BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING
FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF MATHEMATICS
ÚSTAV MATEMATIKY

STOCHASTIC OPTIMIZATION OF NETWORK FLOWS
STOCHASTICKÁ OPTIMALIZACE TOKŮ V SÍTÍCH

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Martin Málek**
AUTOR PRÁCE

SUPERVISOR — **RNDr. Pavel Popela, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2017

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Specification Master's Thesis

Department:	Institute of Mathematics
Student:	**Bc. Martin Málek**
Study programme:	Applied Sciences in Engineering
Study field:	Mathematical Engineering
Leader:	**RNDr. Pavel Popela, Ph.D.**
Academic year:	2016/17

Pursuant to Act no. 111/1998 concerning universities and the BUT study and examination rules, you have been assigned the following topic by the institute director Master's Thesis:

Stochastic Optimization of Network Flows

Concise characteristic of the task:

The student will deepen his knowledge of network flow problems. He will prove the acquired knowledge of graph theory, mathematical programming, probability theory and mathematical statistics. He will focus on models of stochastic programming, and especially, on the issues of modification, decomposition, and implementation of large-scale models. The student will utilize his knowledge of optimization models, logistics and deepen them in relation to the principles of modeling uncertainty. He will study properties of the selected problems, and the selection and modification of the appropriate algorithms will follow. Test calculations will be performed by using real-world data. The problems studied within the framework of NETME + and with Norwegian partners in Molde and Bergen will be also discussed.

Goals Master's Thesis:

1. The theoretical part will present a clear overview of carefully selected ideas from graph theory and mathematical programming in relation to the tasks linked to network flow problems.
2. The author of thesis will focus on building of original models suitable for stochastic network problems in logistics and on analysis of their properties.
3. The author of thesis will select and detail the efficient algorithms (and their modifications) for solving of the selected network flow problems.
4. The thesis will include software implementation of models and algorithms together with results of test calculations.
5. Results achieved for the real-world data will be further presented, analyzed and interpreted.

List of literature:

BIRGE, John R. and François LOUVEAUX. Introduction to Stochastic Programming. Springer Verlag. 1997. ISBN: 978-1-4614-0236-7.

CHRISTOFIDES, Nicos. Graph Theory - an Algorithmic Approach. Academic Press. 1975. ISBN 0-1--174350-0.

WOLSEY, Laurence A. Integer programming. New York: John Wiley & Sons. 1998. ISBN 978-0-4-1-28366-9.

GHIANI, Gianpaolo, LAPORTE, Gilbert and Roberto MUSMANNO. Introduction to logistics systems planning and control. Hoboken, NJ, USA: J. Wiley. 2004. ISBN 0-470-84917-7.

WALLACE, Stein W. and Alan KING. Modeling with Stochastic Programming. Springer Verlag. 2012. ISBN 978-0-387-87816-4.

KALL, Peter and Stein W. WALLACE. Stochastic Programming. New York: John Wiley & Sons. 1993. ISBN 978-0471951582.

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2016/17

In Brno,

L. S.

prof. RNDr. Josef Šlapal, CSc.	doc. Ing. Jaroslav Katolický, Ph.D.
Director of the Institute	FME dean

Abstrakt
Magisterská práce se zabývá stochastickou optimalizací síťových úloh. Teoretická část pokrývá tři témata - teorii grafů, optimalizaci a progressive hedging algoritmus. V rámci optimalice je hlavní část věnována stochastickému programování a dvoustupňovému programování. Progressing hedging algoritmus zahrnuje také metodu přiřazování scénářů a modifikaci obecného algoritmu na dvou stupňové úlohy. Praktická část je věnována modelům na reálných datech z oblasti svozu odpadu v rámci České republiky. Data poskytl Ústav procesního inženýrství.

Summary
The master's thesis focuses on the stochastic optimization in network flow problems. The theoretical part covers three topics - the graph theory, the optimization and the progressive hedging algorithm. Within the optimization the main part is devoted to the stochastic programming and the two-stage programming. The progressive hedging algorithm includes also the scenario aggregation and the modification of the general algorithm to two-stage problems. The practical part deals with models using real-world data of collection of municipal waste within the Czech Republic, which were provided by the Institute of Process Engineering.

Klíčová slova
Teorie grafů, toky v sítích, stochastické programování, dvoustupňová optimalizace, progressive hedging algoritmus

Keywords
Graph theory, network flows, stochastic programming, two-stage optimization, progressive hedging algorithm

MÁLEK, M. *Stochastic Optimization of Network Flows.* Brno: University of technology, Faculty of mechanical engineering, 2017. 82 p. Supervisor RNDr. Pavel Popela, Ph.D.

I declare that I wrote my master's thesis Stochastic Optimization of Network Flows independently under the guidance of RNDr. Pavel Popela, Ph.D. using the materials listed in the literature.

Bc. Martin Málek

I would like to thank my supervisor, RNDr. Pavel Popela, Ph.D., for the professional conduct of my master's thesis and his remarks. I would also like to thank Ing. Radovan Šomplák, Ph.D. as the consultant from the Institute of Process Engineering for provided real world data and the guidance in the Waste Management. My thanks belong to Ing. Jakub Kůdela and Ing. Vlastimír Nevrlý for help both theoretically and practically.

Bc. Martin Málek

Table of Contents

Aims of the Thesis

This work will be built on my bachelor's thesis, Selected Optimization Models in Logistics, in which I dealt with the variety of logistics problems based on flow networks, for instance the models with capacity constraints, dynamic price problems, finding optimal locations for incinerator construction and others. All the parameters and decisions made by the decision maker were deterministic, i.e. they did not contain the randomness inside.

In comparison with bachelor's thesis I will go further and in my master's thesis, Stochastic Optimization of Network Flows, I will focus on stochastic optimization of flow networks, which includes the randomness. Mainly, I will deal with the two-stage problems, where the decision maker makes two decisions in two different moments. I will challenge also mixed integer problems and an alternative way of computing will be provided for such problems in a form of progressive hedging algorithm. The modification of progressive hedging algorithm will be discused for two-stage and for mixed integer problems.

Furthermore the models will be computed using real world data of a collection of a municipal waste within the Czech Republic, which were provided by the Institute of Process Engineering. In the thesis, I use the software GAMS for the optimization, MS Excel to work with input and output data, C++ in Visual Studio to program the progressive hedging algorithm and optimization software AIMMS to model graphical results.

Preface

The theory is written in the first three chapters. The last chapter focuses on an application. The thesis begins with the Chapter 1 devoted to the Graph Theory. Basic concepts of graphs, trees, flow networks and other basic notions are provided and are thoroughly illustrated on figures for better understanding. At the end of the chapter, adjacency and incidence matrices are explained. These two concepts create a connection between the theory and the application.

The Chapter 2 Optimization is divided into three sections. The first Section 2.1 contains a definition of the deterministic problem and gives the elementary overview of linear programming, non-linear programming and mixed-integer programming. Next sections create the major part of this chapter. The Section 2.2 Stochastic Programming brings the new concept of how to build models containing the randomness inside. According to the moment, when the randomness is revealed, we distinguish two approaches, here-and-now approach and wait-and-see approach. For the mathematical correct way of formulation, deterministic reformulations and comparisons are also given. The Section 2.3 Multi-stage Stochastic Programming deals with the problems, in which the decision in more than one time moment is needed. Starting up with two-stage problems, it is generalized into the multi-stage problems.

The Chapter 3 Progressive Hedging Algorithm brings an alternative way of a computation stochastic problems. It consists of two sections. In the first Section 3.1 Scenarios Aggregation, the theory is described and illustrated on an example of partitioning the scenario set. The second Section 3.2 Progressive Hedging Algorithm for Multi-stage Stochastic Programming focuses on the general algorithm for multi-stages problems, which is then restricted to the case of two-stage problems.

The Chapter 4 Real-Data Applications contains 3 models as a result of previous theoretical chapters. The first model 4.1 Large-scale Problem is applied to the real data of the collection of municipal waste of municipalities with extended powers within the Czech Republic. The model is using the probability weight and simulations to determine the behaviour of the two methods how to deal with the waste - the waste utilization and the waste disposal. The second model is the well-known two-stage Farmer's Problem 4.2, which is used for the implementation of the progressive hedging algorithm. Results are compared to the computation with and without the progressive hedging algorithm. The last Section 4.3 describes the Main Problem as a sequel of my bachelor's thesis. It is the two-stage model with the binary decision whether to build or not additional incinerators. The model is firstly tested on a small network and then implemented to the real data that were used in the Large-scale Problem. The problem is computed using the classical solvers and then using the progressive hedging algorithm. At the end, the sequence of graphical results is given.

1. Graph Theory

In this chapter I will provide some fundamental concepts from the Graph theory, which I will mainly use in later chapters. As a motivation, see the chapter 4.

Among the first historical mentions belongs the Euler's *Königsberg bridges problem* (1735) (see [3]). Since this historical time the Graph theory has begun. Graph theory is a basic tool for visualizing a real world situations. We can easily represent entities by points and relations between them by lines to create a graphical figure called *graph*. Depending on the branch of our interest graphs can be found under different names such as sociograms, organization structures, networks, ...
(see [1, 2, 3]).

A *graph* G is a collection of points $v_1, v_2, ..., v_n$ called *vertices* and a collection of lines $e_1, e_2, ..., e_m$ called *edges*. A graph is described as a pair $G = (V, E)$, where V is a set of vertices and E is a set of edges.

In this case we are talking about an *undirected graph*. A *directed graph* is a pair $G = (N, A)$, where N represents a set of *nodes* and A is a collection of *arcs*. Each arc connects two nodes $a = (i, j)$, $i, j \in N$. The initial node of an arc is called a *tail* and the final node is a *head* or we can say, that an arc is *incident to* node i and j. Term *indegree* represent a number of arcs incoming to the node and *outdegree* is a number of arcs outcoming from the node. Now we can formulate *degree of node* as:

$$degree\ of\ node = indegree + outdegree$$

If there exists a single arc connecting nodes i and j, $\forall i, j \in N$, then we are talking about *simple directed graph*. Else the number of arcs from i to j refers to *multiplicity* of an arc.

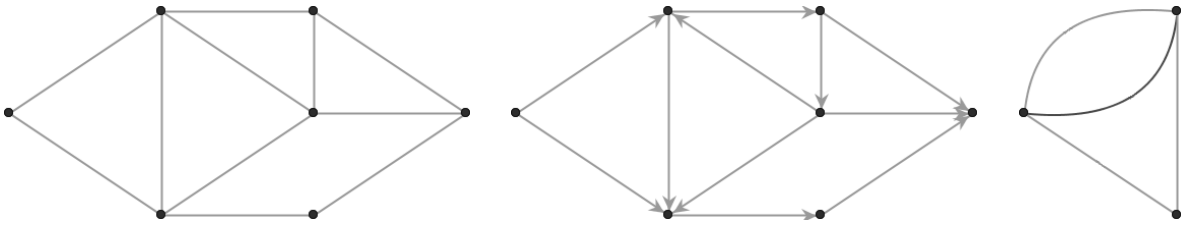

Figure 1.1: Simple undirected graph, simple directed graph and multigraph.

A graph $G' = (N', A')$ is a *subgraph* of the graph $G = (N, A)$, if the set of nodes $N' \subseteq N$ and the set of arcs $A' \subseteq A$. If $N' = N$ and $A' \subseteq A$, then G' is a *spanning subgraph*. A *walk* is a subgraph of directed G. The walk is a node-arc sequence $i_1 - a_1 - i_2 - a_2 - ... - a_{n-1} - i_n$ where each arc has its tail and head. A *path* is a walk without repetitions.

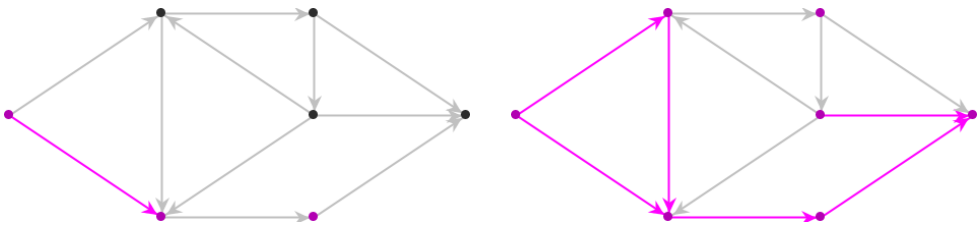

Figure 1.2: Subgraph and spanning subgraph.

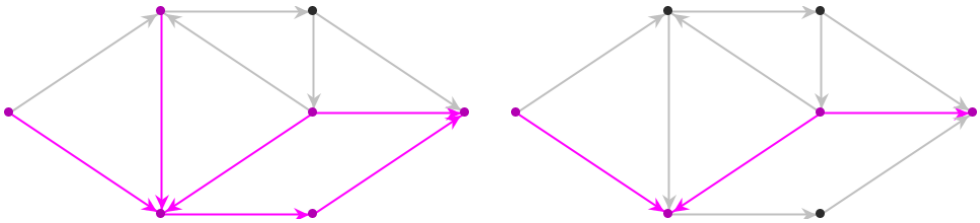

Figure 1.3: Walk and path.

Nodes i and j are *connected* if the graph G has at least one path from i to j. A *cycle* is represented as a path, when the first node is connected to the last one. A *tree* is connected graph with no cycle satisfying following assumptions:

- A tree containing n nodes has exactly $n-1$ arcs.
- A tree has at least two nodes with degree one.
- Every two nodes of a tree are connected by a unique path.

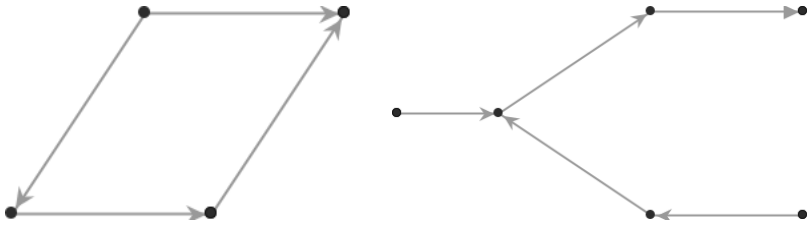

Figure 1.4: Cycle and tree.

We can give an attribute to an arc or a node. Among the common attributes belong *weights, lenghts, costs, ...* Assigning a weight to an arc we are talking about *arc-weighted* graph. A weight is given by a *weight function* $w : A(G) \to \mathbf{R}^+$. The symbol $A(G)$ represents the set of arcs of the graph G. In the similar way a node with an attribute is called *node-weighted*. If the graph has both, then it is called *weighted*. In the network flow theory there are two distinct nodes with special properties - *source* s and *sink* t.

- Source has indegree $= 0$.
- Sink has outdegree $= 0$.

A *flow network* is represented as a quartet (G, s, t, w). A *flow* $S = (G, s, t, w)$ is a mapping $x : A(G) \to \mathbf{R}$ satisfying following conditions:

- *Capacity constraint:* The flow $x(a), a \in A$ is assuming to be nonnegative and cannot exceed capacity *w(a)*:
$$\forall a \in A(G) : 0 \leq x(a) \leq w(a).$$
- *Flow conservation:* An inflow to node n equals to an outflow from node n:
$$\forall n \in N(G) - \{s, t\} : \sum_{a \in A_G^-(n)} x(a) = \sum_{a \in A_G^+(n)} x(a),$$

where $A_G^-(n)$ represents the set of all arcs for which the final node is n and $A_G^+(n)$ represents the set of all arcs for which the initial node is n.

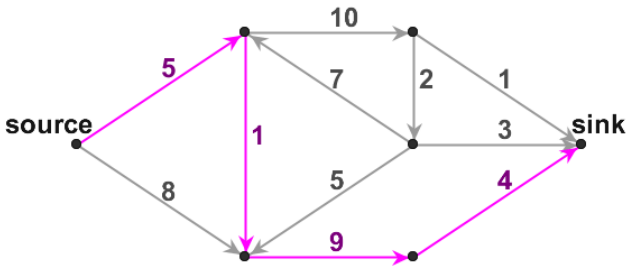

Figure 1.5: Flow network.

In the figure 1.5 you can see an arc-weighted flow network with a possible flow from the source to the sink represented as a directed path.

An algebraic method of representing a graph is by use of matrices. I will provide two basic ways - by the *adjacency matrix* and by the *incidence matrix*. For better understanding I will asume the following network (see [1]).

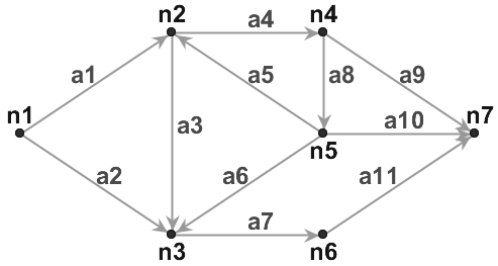

Figure 1.6: General flow network.

The *adjacency matrix* $A = [a_{ij}]$ is $n \times n$ matrix for a graph G of n nodes and it is defined by:

$$a_{ij} = 1 \text{ if arc } (n_i, n_j) \in A(G)$$
$$a_{ij} = 0 \text{ if arc } (n_i, n_j) \notin A(G)$$

The adjacency matrix shown in the figure 1.6 is:

	n_1	n_2	n_3	n_4	n_5	n_6	n_7
n_1	0	1	1	0	0	0	0
n_2	0	0	1	1	0	0	0
n_3	0	0	0	0	0	1	0
n_4	0	0	0	0	1	0	1
n_5	0	1	1	0	0	0	1
n_6	0	0	0	0	0	0	1
n_7	0	0	0	0	0	0	0

The sum of all elements in the row n_i gives us the outdegree of the node n_i and the sum of all elements in the column n_i gives the indegree of the node n_i.

The *incidence matrix* $M = [m_{ij}]$ is $n \times m$ matrix for a graph G of n nodes and m arcs and it is defined by:

$$m_{ij} = \ 1 \text{ if node } n_i \text{ is the tail of arc } a_j$$
$$m_{ij} = -1 \text{ if node } n_i \text{ is the head of arc } a_j$$

The incidence matrix shown in the figure 1.6 has the following form:

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}
n_1	1	1	0	0	0	0	0	0	0	0	0
n_2	−1	0	1	1	−1	0	0	0	0	0	0
n_3	0	−1	−1	0	0	−1	1	0	0	0	0
n_4	0	0	0	−1	0	0	0	1	1	0	0
n_5	0	0	0	0	1	1	0	−1	0	1	0
n_6	0	0	0	0	0	0	−1	0	0	0	1
n_7	0	0	0	0	0	0	0	0	−1	−1	−1

Each arc is incident to node i and j so the sum of all elements in the column a_i is 0.

For further information about the flow network see [4] and for the direct application of graph theory see Chapter 4, especially the Section 4.3.

2. Optimization

This chapter highlights some basic concepts from optimization. Starting with the notion of mathematical programming, linear programming, nonlinear programming and mixed-integer programming it will continue with stochastic programming and the key differences comparing to the deterministic programming. The chapter will be mainly focused on stochastic programming. The notions of wait-and-see and here-and-now approaches will be discussed and the deterministic reformulations will be also given. Furthermore, the two-stage stochastic programming will be noted and from that it will be generalized to the multi-stage stochastic programming.

First, let me introduce to you some historical facts. Optimization has been greatly developed during and after the World War II as well as the Operations research. In 1947 G. Dantzig presented the Simplex method for solving Linear programming problems, which were highly used for the theory. J. Von Neumann established the theory of duality for the linear problems. In 1951 H. W. Kuhn and A. W. Tucker invented Optimality conditions for Nonlinear programming and after that R. T. Rockafellar and others developed the duality and the theory behind. In the same year H. Markowitz presented his Portfolio theory based on quadratic optimization and in 1990 received in memoriam the Nobel prize in economics. The Network Flow approach started in 1950 by Merril Flood and was further developed by Ford and Fulkerson in 1954. Few years later Network Flow was connected to the Graph Theory. In 1958 R. Gomory came up with Integer programming and developed the "cutting" plane method which was proven as very effective with Branch and bound method. Since 1955 the Stochastic programming has begun with the publication "Linear Programming under Uncertainty". In 1960s R. Wets and in 1980s J. Birge extended the approach. In those years computers became more efficient. Heurestic algorithms for global optimization and large scale problems came to the light of interest and with them the algorithms. Among the most famous algorithms belongs the Progressive Hedging Algorithm or the Benders decomposition. See [15].

2.1. Deterministic Programming

A mathematical model is called *deterministic* if all parameters are assumed to be known with certainity, i.e. all coefficients in the objective function and in constraints are known. The underlying concept of optimization is the Mathematical Programming.

2.1.1. Mathematical Programming

According to Ronald L. Rardin, [5], any problem can be modeled by the three dimensions:

- the *decisions* open to decision makers,
- the *constraints* limiting decision choices,
- and the *objectives* making some decisions preferred to others.

These three dimensions formulate the *mathematical program* (MP) (or *optimization model*) representing problem choices as decision variables. The MP is composed of *objective functions*, *constraints* and *bounds*. Our goal is to maximize or minimize values of objective functions subject to constraints expressing the limits on possible decisions. A *feasible set* is a set of possible values for decision variables that satisfies all constraints. The MP is modeled as follows:

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in C, \end{aligned} \tag{2.1}$$

where $C \subseteq \mathbf{R}^n$ is a feasible set, $n \in \mathbf{N}$. $f : C \to \mathbf{R}$ is an objective function and $\mathbf{x} \in C$ represents a decision variable.

The *optimal solution* is a feasible solution better than any other feasible solution. It will be denoted by $\mathbf{x}^*$.

2.1.2. Linear Programming

MP (2.1) is called *linear program* (LP), if the objective function f and all the constraints are linear in the decision variables, [5]. The linear program is written as:

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \Diamond \mathbf{b}, \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \end{aligned} \tag{2.2}$$

where the objective function $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$ consists of decision variables $\mathbf{x} = (x_j)_{j\in\mathcal{J}} \in \mathbf{R}^n$. $\mathcal{J} = \{1, ..., n\}$ represents the set of indices of variables. The vector $\mathbf{c} = (c_j)_{j\in\mathcal{J}} \in \mathbf{R}^n$ contains coefficients of the objective function. $\mathcal{I} = \{1, ..., m\}$ is the set of indices of constraints. The vector $\mathbf{b} = (b_i)_{i\in\mathcal{I}} \in \mathbf{R}^m$ and the matrix $\mathbf{A} = (a_{ij})_{i\in\mathcal{I},j\in\mathcal{J}} \in \mathbf{R}^{m\times n}$ are coefficients of constraints. Vectors $\mathbf{l} = (l_j)_{j\in\mathcal{J}} \in \mathbf{R}^n$ and $\mathbf{u} = (u_j)_{j\in\mathcal{J}} \in \mathbf{R}^n$ are lower and upper bounds of the LP. The symbol $\Diamond = (\diamond_i)_{i\in\mathcal{I}} \in \{\leq, =, \geq\}^m$ expresses relations between the right hand side and the left hand side in constraints, [7].

LP (2.2) is the special case of (2.1). It is the common model for network flow problems. Sometimes, we want to underline some deterministic parameter. For this purpose the *parametric* (PMP) is useful:

$$\begin{aligned} \min \quad & f(\mathbf{x}, \mathbf{a}) \\ \text{s.t.} \quad & \mathbf{x} \in C(\mathbf{a}), \end{aligned} \tag{2.3}$$

where $\mathbf{a} \in \mathbf{R}^K$, $K \in \mathbf{N}$ is a *constant parameter*. This model will be useful for the next chapters, where the constant parameter will represent uncertainities.

2.1.3. Nonlinear Programming

MP (2.1) is called a *nonlinear program* (NLP), if the objective function f or any constraint is nonlinear in the decision variables, [5]. NLP is defined in the following way:

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \Diamond \mathbf{0}, \\ & \mathbf{x} \in C, \end{aligned} \tag{2.4}$$

where $\mathbf{g} : \mathbf{R}^n \to \mathbf{R}^m$ is nonlinear and $C \subset \mathbf{R}^n$, [7].

2.1.4. Mixed-Integer Programming

So far we have mentioned the case of the continuous decision variable. The decision variable can have an *integer* or *discrete* character, if it is limited to a countable set of values, [5]. In the later chapters, the binary decision variable will be presented. The binary variable is frequently used for logical conditions.

$$x_i = \begin{cases} 1 & \text{if the i-th component of } \mathbf{x} \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

We obtain a *mixed-integer program* (MIP) from the MP (2.1), if any of decision variables are discrete. The linear MIP can be written as:

$$\begin{aligned} \min \quad & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \Diamond \mathbf{b}, \\ & \mathbf{x} \in \mathcal{D}, \end{aligned} \tag{2.5}$$

where $\mathcal{D} \subset \mathcal{Z}^l \times \mathbf{R}^{n-l}$. The set of all integers is represented by $\mathcal{Z}$, see [7].

2.2. Stochastic Programming

A mathematical model is termed *probabilistic* or *stochastic*, (SP), if it involves quantities known only in probability [5]. Randomness is given by ($\boldsymbol{\xi}$) symbol following the parameters denoting them. SP can be obtained by PMP (2.3) by replacing some of the parameters with random parameters:

$$\begin{aligned} \min \quad & f(\mathbf{x}, \boldsymbol{\xi}) \\ \text{s.t.} \quad & \mathbf{x} \in C(\boldsymbol{\xi}), \quad \text{a.s.} \end{aligned} \tag{2.6}$$

where $\boldsymbol{\xi} : \Omega \to \mathbf{R}^K$ is a *random vector* for given $(\Omega, \mathcal{F}, \mathcal{P})$ *probability space*. We assume that the family $\mathcal{F} \subset \Omega$ of *events* and the *probability distribution* $\mathcal{P}$ are known. Hence for every subset $A \subset \Omega$ that is an event, i.e. $A \in \mathcal{F}$, the probability $\mathcal{P}(A)$ is known, see [6, 7]. A *realization* ω^s of $\boldsymbol{\xi}$ is defined as $\forall \omega^s \in \Omega : \boldsymbol{\xi}(\omega^s) \in \mathbf{R}^K$. I will use a shortcut $\boldsymbol{\xi}^s$. But what is the meaning of (2.6) before the realization $\boldsymbol{\xi}^s$ is observed? If I take a decision $\mathbf{x}$ before knowing the realization $\boldsymbol{\xi}^s$, the program (2.6) is not well-defined. This leads to so-called *deterministic equivalents* or *deterministic reformulations*.

According to the moment when the observation is made, we distinguish two main approaches: The *wait-and-see approach* (WS) and the *here-and-now approach* (HN), see figure 2.2.1.

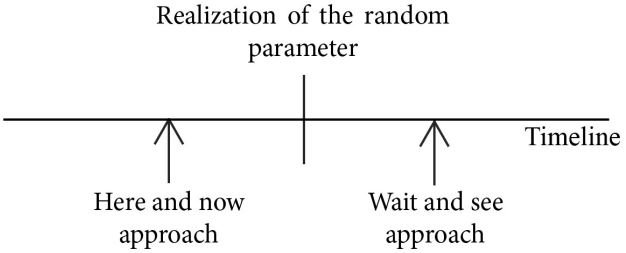

Figure 2.2.1: Graphical representation of WS and HN approaches.

2.2.1. Wait and See Approach

WS approach discusses a situation after the observation $\boldsymbol{\xi}^s$ is made. Hence, the decision variable $\mathbf{x}$ is the response on $\boldsymbol{\xi}$ and therefore $\mathbf{x}$ is the function of $\boldsymbol{\xi}$, i.e. $\mathbf{x}(\boldsymbol{\xi})$. In other words, we know how the future will look like and *"our problem under uncertinty is solved by a decision under certainty"* [6]. Also the function of outcome $f(\mathbf{x}(\boldsymbol{\xi}), \boldsymbol{\xi})$ depends on random variable. WS approach is useful for long-term plannig. The model can be written as:

$$\begin{aligned} \min \quad & f(\mathbf{x}(\boldsymbol{\xi}), \boldsymbol{\xi}) \\ \text{s.t.} \quad & \mathbf{x}(\boldsymbol{\xi}) \in C(\boldsymbol{\xi}) \quad \text{a.s.} \end{aligned} \tag{2.7}$$

The solution will be denoted by $\mathbf{x}_{\min}^{\mathrm{WS}}(\boldsymbol{\xi})$ and the value of objective function by $z_{\min}^{\mathrm{WS}}(\boldsymbol{\xi})$. I will use an abbreviation in notation $\mathbf{x}_{\min}^{\mathrm{WS}}$ and $z_{\min}^{\mathrm{WS}}$.

2.2.2. Here and Now Approach

More common situation in a real world problem is when we have to make a decision and yet we do not know realizations of random vector $\boldsymbol{\xi}$. Hence the decision $\mathbf{x}$ is the same for any future realization of $\boldsymbol{\xi}$. This approach is called here-and-now and is modelled as:

$$\begin{aligned} \min \quad & f(\mathbf{x}, \boldsymbol{\xi}) \\ \text{s.t.} \quad & \mathbf{x} \in C(\boldsymbol{\xi}) \quad \text{a.s.} \end{aligned} \tag{2.8}$$

The solution will be denoted by $\mathbf{x}_{\min}^{\mathrm{HN}}$ and the value of objective function by $z_{\min}^{\mathrm{HN}}$.
Several questions arise:

1. What is the meaning of model (2.8) before the $\boldsymbol{\xi}^s$?
2. What if is there more than one decision to make?

The questions will be answered in the following sections.

2.2.3. Deterministic Reformulations

To get a well-defined model (2.8) we use so-called *deterministic reformulations* or *deterministic equivalents*. In contrast with WS approach, we have several different possibilities, how to define HN approach. I will provide an insight into the *individual scenario* (IS), *expected value* (EV), *expected objective* (EO) approaches and comparisons between SP - *the value of stochastic solution* (VSS) and *expected value of perfect information* (EVPI).

Individual Scenario Deterministic Reformulation

The first idea for the HN can be by applying one of known WS solutions for certain $\boldsymbol{\xi}^s$, where realizations $\boldsymbol{\xi}^s$ are called *scenarios*. The individual scenario consists of picking up one realization and then solving the problem by the method of WS. The program has a form:

$$\begin{aligned} \min \quad & f(\mathbf{x}, \boldsymbol{\xi}^s) \\ \text{s.t.} \quad & \mathbf{x} \in C(\boldsymbol{\xi}^s), \end{aligned} \tag{2.9}$$

where $\boldsymbol{\xi}^s \in \Omega$ is a chosen individual scenario. The solution will be denoted by $\mathbf{x}^{\text{IS}}_{\min}$ and the value of objective function by $z^{\text{IS}}_{\min}$. To have a feedback how good our solution is we can compute $\zeta^{\text{IS}}_s = f(\mathbf{x}^s_{\min}, \boldsymbol{\xi}), \quad \forall s \in S$ and obtain a vector of random variables. The $\mathbf{x}^s_{\min}$ represents a scenario solution. The probability distributions of ζ^{IS}_s describes *future costs* in case of implementig such solution. We can also compute the expected value $E\left[\zeta^{\text{IS}}_s\right]$ for some $s \in S$ or *risk awareness* by computing the variance

$$var\left[\zeta^{\text{IS}}_s\right] = E\left[(\zeta^{\text{IS}}_s)^2\right] - E\left[\zeta^{\text{IS}}_s\right]^2$$

for some $s \in S$. IS approach is widely used in economics.

Expected Value Deterministic Reformulation

We do not might be such strict to choose just one individual scenario. We can replace $\boldsymbol{\xi}$ with convex combination or weighted average. This is done by computing $E[\boldsymbol{\xi}]$. Such program is called expected value and its reformulation has a form:

$$\begin{aligned} \min \quad & f(\mathbf{x}, E[\boldsymbol{\xi}]) \\ \text{s.t.} \quad & \mathbf{x} \in C(E[\boldsymbol{\xi}]), \end{aligned} \tag{2.10}$$

where $E[\boldsymbol{\xi}]$ is an expected value of $\boldsymbol{\xi}$. The solution will be denoted by $\mathbf{x}^{\text{EV}}_{\min}$ and the value of objective function by $z^{\text{EV}}_{\min}$. As in IS case we can substitute the solution into $\zeta^{\text{EV}} = f(\mathbf{x}^{\text{EV}}_{\min}, \boldsymbol{\xi})$ and from this compute $E\left[\zeta^{\text{EV}}\right]$ interpreted as *average costs* and $var\left[\zeta^{\text{EV}}\right]$.

Comparison of IS and EV solutions

For EV deterministic reformulation we define the *Expected objective function value for the optimal solution of EV (EEV)* as

$$EEV = E_\xi\left[f(\mathbf{x}^{\text{EV}}_{\min}, \boldsymbol{\xi})\right] = E\left[\zeta^{\text{EV}}\right].$$

Subtracting $z^{\mathrm{EV}}_{\min}$ from the EEV characteristic we get an information whether $z^{\mathrm{EV}}_{\min}$ *looks optimistic.*

$$EEV - z^{\mathrm{EV}}_{\min}$$

Computing the variances $var\left[\zeta^{\mathrm{EV}}\right]$ and $var\left[\zeta^{\mathrm{IS}}_s\right]$ we get a tool for comparing the IS and EV solutions.

Expected Objective Deterministic Reformulation

The idea of expected objective is to take an expected value $E\left[f(\mathbf{x}, \boldsymbol{\xi})\right]$ of the objective function. The EO corresponds to the HN approach and its deterministic reformulation has the following form:

$$\begin{aligned} \min \quad & E\left[f(\mathbf{x}, \boldsymbol{\xi})\right] \\ \text{s.t.} \quad & \mathbf{x} \in \mathbf{R}^n \end{aligned} \tag{2.11}$$

The EO program (2.11) is unconstrained. The solution will be denoted by $\mathbf{x}^{\mathrm{EO}}_{\min}$ and the value of objective function by $z^{\mathrm{EO}}_{\min}$.

Comparisons between SP

- *EV and EO*

  In general, we compare $E\left[f(\mathbf{x}, \boldsymbol{\xi})\right]$ with $f(\mathbf{x}, E\left[\boldsymbol{\xi}\right])$ making the difference:

  $$E\left[f(\mathbf{x}, \boldsymbol{\xi})\right] - f(\mathbf{x}, E\left[\boldsymbol{\xi}\right]) = E\left[\boldsymbol{\xi}^2\right] - \left(E\left[\boldsymbol{\xi}\right]\right)^2 = var[\boldsymbol{\xi}] \geq 0.$$

  From this we get the first result:

  $$E\left[f(\mathbf{x}, \boldsymbol{\xi})\right] \geq f(\mathbf{x}, E\left[\boldsymbol{\xi}\right]).$$

  For conclusion we need to define the *Jensen's inequality*: If $f(\mathbf{x}, \boldsymbol{\xi})$ is a convex function at $\boldsymbol{\xi}$, then $E\left[f(\mathbf{x}, \boldsymbol{\xi})\right] \geq f(\mathbf{x}, E\left[\boldsymbol{\xi}\right])$.

- *EO and EEV*

  We define the VVS, *value of stochastic solution*, as:

  $$VSS = EEV - z^{\mathrm{EO}}_{\min}.$$

  "VSS characteristic measures how much can be saved when the true HN approach is used instead of EV approach"[7].

- *WS and HN*

  Expected value of perfect information, EVPI characteristic is defined as:

  $$EVPI = z^{\mathrm{EO}}_{\min} - E\left[z^{\mathrm{WS}}_{\min}(\boldsymbol{\xi})\right].$$

  EVPI compares the EO with WS and it can be perceived as *the amount which we are willing to pay in order to know the future realizations of randomness.*

2.3. Multi-Stage Stochastic Programming

Multi-stage SP represents an idea how to modify our MP if we have more than one decision to make. I will begin with two decisions, see Subsection 2.3.1, then in the Subsection 2.3.2 I will provide the MP model that can solve my problem and at the end in the Subsection 2.3.3 I will consider generalized issue about the number of decisions.

2.3.1. Two-Stage Stochastic Programming

Let us start with the meaning of the *stage*. The stage is a time interval in which we take one decision. Depending on the time of realization $\boldsymbol{\xi}$, we can partition the set of all decisions into the two following subsets:

- Decisions $\mathbf{x}$ that have to be taken without full information about some random events
- Decisions or *corrective actions* $\mathbf{y}$ taken after the realization of $\boldsymbol{\xi}$.

In our case, we are dealing with two decisions (two stages): $\mathbf{x}$ and $\mathbf{y}$. The decision $\mathbf{x}$ taken before the realization $\boldsymbol{\xi}$ is called the *first-stage decision* and this time period first-stage. Now, the randomness is revealed followed by the *second-stage decision* $\mathbf{y}$ and the time period called the *second-stage.* We can illustrate this process as:

$$\mathbf{x} \rightarrow \boldsymbol{\xi}(\omega) \rightarrow \mathbf{y}(\omega),$$

where $\mathbf{y}(\omega)$ can often depend on $\mathbf{x}$, i.e $\mathbf{y}(\omega, x)$, see [8]. The next Subsection 2.3.2 shows how to model such situations.

2.3.2. Two-Stage Linear Stochastic Programming with Fixed Recourse

Two-stage linear stochastic program with fixed recourse can be modelled as follows:

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} + E_{\boldsymbol{\xi}}\left[\min \quad \mathbf{q}^T(\omega)\mathbf{y}(\omega)\right] \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{T}(\omega)\mathbf{x} + \mathbf{W}\mathbf{y}(\omega) = \mathbf{h}(\omega), \qquad \text{a.s.} \\ & \mathbf{x} \geq \mathbf{0}, \mathbf{y}(\omega) \geq \mathbf{0}, \qquad \text{a.s.} \end{aligned} \tag{2.12}$$

where $\mathbf{x}$ represents first-stage decisions and $\mathbf{A}$, $\mathbf{c}$ and $\mathbf{b}$ belong to the first-stage. For a given realization ω we have the second-stage data $\mathbf{q}(\omega)$, $\mathbf{T}(\omega)$, $\mathbf{h}(\omega)$ and a decision $\mathbf{y}(\omega)$. The two-stage stochastic program is said to have a *fixed recourse* if the matrix $\mathbf{W}$ does not depend on randomness, i.e. it is fixed, see [6, 8]. The dependence $\mathbf{y}$ on ω indicates that the decisions are different for various realizations of ω. They are chosen in such way that the constraints $\mathbf{T}(\omega)\mathbf{x} + \mathbf{W}\mathbf{y}(\omega) = \mathbf{h}(\omega)$ and the bound $\mathbf{y}(\omega) \geq \mathbf{0}$ hold almost surely (we use abbreviation a.s.). This means that they hold $\forall\omega \in \Omega$ except the sets with zero measure (probability).

2.3.3. Multi-Stage Stochastic Programming

In the previous Subsection 2.3.2 I assumed only two decisions to make $\mathbf{x}$ and $\mathbf{y}(\omega)$. But most practical decisions problems involve more than just two decisions. I can consider decisions as a sequence that react to outcomes developed over time. Each decision is taken in different stage (period). As an ilustrative example see figure 2.3.1. The multi-stage linear program with fixed recourse has a form:

$$\begin{aligned} \min \ & \mathbf{c}^{1^T}\mathbf{x}^1 + E_{\boldsymbol{\xi}^2}\left[\min \quad \mathbf{c}^{2^T}(\omega)\mathbf{x}^2(\omega)\right] + ... + E_{\boldsymbol{\xi}^H}\left[\min \quad \mathbf{c}^{H^T}(\omega)\mathbf{x}^H(\omega)\right] \qquad (2.13) \\ \text{s.t.} \quad & \mathbf{W}^1\mathbf{x}^1 = \mathbf{h}^1, \\ & \mathbf{T}^1(\omega)\mathbf{x}^1 + \mathbf{W}^2\mathbf{y}^2(\omega) = \mathbf{h}^2(\omega), \quad a.s. \\ & \vdots \\ & \mathbf{T}^{H-1}(\omega)\mathbf{x}^{H-1} + \mathbf{W}^H\mathbf{y}^H(\omega) = \mathbf{h}^H(\omega), \quad a.s. \\ & \mathbf{x}^1 \geq \mathbf{0}, \mathbf{x}^t(\omega) \geq \mathbf{0}, \quad a.s., t = 2, ..., H, \end{aligned}$$

where all constraints hold a.s. $\mathbf{c}^1$ and $\mathbf{h}^1$ are known vectors. Stages are denoted by index $t = 1, ..., H$. The last stage is labeled as H. The multi-stage model is similar to the model (2.12).

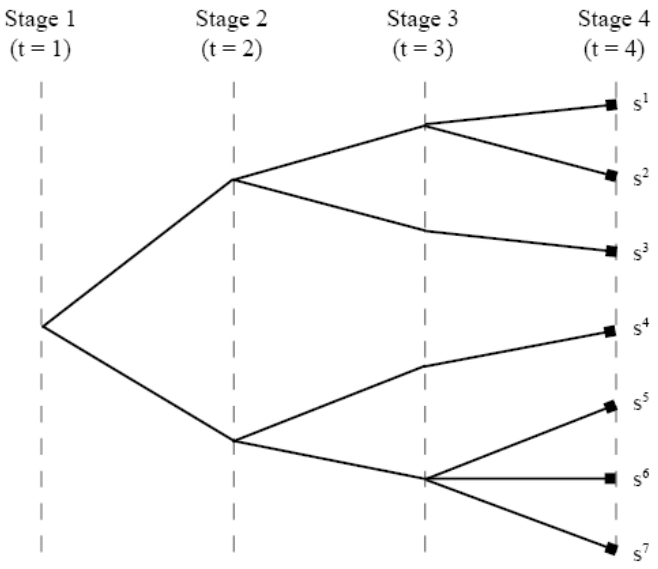

Figure 2.3.1: A tree of seven scenarios over four periods.

The tree is often used for description of scenarios such as in the figure 2.3.1. There are four stages ($t = 1, ..., 4$). In the last stage ($H = 4$) we have seven scenarios. In the previous stages ($t < 4$), we have a more limited number of possible realizations, which we call the *stage t scenarios*. Each of these period t scenarios is said to have a single *ancestor* scenario in stage $(t - 1)$ and perhaps several *descendant* scenarios in stage $(t + 1)$. We note that different scenarios at stage t may correspond to the same $\boldsymbol{\xi}$ realizations and are only distinguished by differences in their ancestors.

For modelling the multi-stage problem, the *nonanticipativity constraints* are helpful. They state, that decisions taken at any stage of the process do not depend on future decisions. We can mathematically write this as follows:

$$\mathbf{x}(t) = \mathbf{x}(t+1)$$

For further information see [8, 6, 14].

3. Progressive Hedging Algorithm

Stochastic programs arise in a variety of situations. Examples can be found in server location, electricity generation, supply chain design, network and so on. The structure of such programs (for instance combinatorial, nonlinear behaviour) makes them, in general, difficult to solve. We can however obtain an easier solution using a special structure.

A common approach how to represent uncertainty is to formulate a finite number of discrete scenarios associated with probabilities for the values of uncertain parameters. Decisions depend on the number of stages according to which parameter values are assumed to be known to the decision-maker and when the decisions must be made. If we combine the constraints of our problem with an objective to minimize expected cost (by "cost" we can mean also some measure of risk), the resulting SP becomes very large.

The progressive hedging algorithm (PHA) represents an effective method for solving multistage SP. This chapter is divided into two parts, the first one focuses on scenario aggregation 3.1 and the second one deals with the PHA 3.2.

3.1. Scenarios Aggregation

By the term *scenario analysis* we mean to model the uncertainty parameters of our problem by a small number of subproblems derived from the underlying optimization problem 2.6. The key idea is to study the different subproblems and their optimal solutions if there are some similarities or trends or even if we come up with well hedged solution to the underlying problem.

We suppose to deal with time periods (stages) $t = 1, ..., T$. The vector of decisions $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_T) \in \underbrace{\mathbf{R}^{n_1} \times ... \times \mathbf{R}^{n_T}}_{T}$, where $n_1 + ... + n_T = n$. The component $\mathbf{x}_t$ represents the decision which must be made at time t. Furthermore, given a finite set of scenarios $S = \{s^j : j = 1, ..., N\}$ we can formulate the scenario subproblems $(\mathcal{P}_s)$ as:

$$\begin{aligned} \min \quad & f_s(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in C_s, \end{aligned} \tag{3.1}$$

where $f_s(\mathbf{x})$ is the objective function and C_s the feasible set. From the Sections 2.2 and 2.3 we know, how to solve each subproblem individually. Our aim is to tell how to work with the s-dependent solution vectors $\mathbf{x}^s$ and how to obtain an *overall decision* or *decision policy*.

The *policy* is the mapping $\mathbf{X} : S \to \underbrace{\mathbf{R}^{n_1} \times ... \times \mathbf{R}^{n_T}}_{T}$ assigning to each $s \in S$ a vector $\mathbf{X}(s) = (\mathbf{x}_1(s), ..., \mathbf{x}_T(s))$. The component $\mathbf{x}_t(s)$ denotes the decision to be made at time t in the scenario s. The policy has to satify the following constraint: *If two different scenarios s and s' are indistinguishable at time t, then* $\mathbf{x}_t(s) = \mathbf{x}_t(s')$. We can model this constraint by partitioning the scenario set S at each stage t into finitely many disjoint subsets called *scenario bundles*. The scenarios in each bundle are indistinguishable up to time t (the decision $\mathbf{x}_t(\cdot)$ depends only on the information available at time t).

We denote the collection of all scenario bundles at time t as $\mathcal{A}_t$. Then the decision $\mathbf{x}_t(s)$ must be constant relative to scenario s in the bundle A, $s \in A$, for each $A \in \mathcal{A}_t$. The following example will make clear how we can obtain scenario bundles.

Example: Partitioning the scenario set

Let us assume the problem shown in the figure 2.3.1. (Similar example can be found in [11, 12].)

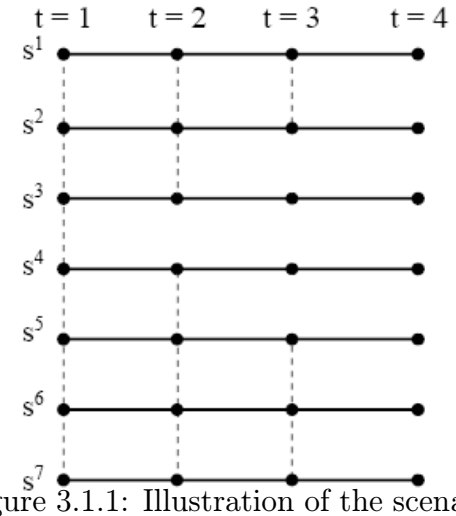

Figure 3.1.1: Illustration of the scenario decomposition scheme.

$\mathcal{A}_1$		$\mathcal{A}_2$		$\mathcal{A}_3$		$\mathcal{A}_4$	
s^1		s^1		s^1	A_1	s^1	A_1
s^2		s^2	A_1	s^2		s^2	A_2
s^3		s^3		s^3	A_2	s^3	A_3
s^4	A_1	s^4		s^4	A_3	s^4	A_4
s^5		s^5		s^5		s^5	A_5
s^6		s^6	A_2	s^6	A_4	s^6	A_6
s^7		s^7		s^7		s^7	A_7

Table 3.1.1: Scenario partitioning using scenario bundles.

In the table 3.1.1 you can see the sequence of collection $\mathcal{A}_t$ at each time period $t = 1, ..., 4$. This table represents partitioning of the scenario set shown in figure 2.3.1. Starting at stage 1, the decision maker does not know any information which could help him with the decision, so the first decision $\mathbf{x}_1(s)$ is the same for all possible scenarios s^i, $i = 1, .., 7$. Hence the first decision does not depend on s. Let us denote this decision as $\boldsymbol{\alpha}$, which is the first element of the policy $\mathbf{X}(s)$.

$$\mathbf{X}(s) = (\boldsymbol{\alpha}, \cdot, \cdot, \cdot) \quad s \in \{s^1, s^2, s^3, s^4, s^5, s^6, s^7\}$$

After the first decision we obtain new information which we can use at the second stage where we have two scenario bundles $A_1 = \{s^1, s^2, s^3\}$ and $A_2 = \{s^4, s^5, s^6, s^7\}$. The scenarios s^1, s^2, s^3 in the bundle A_1 are undistinguishable at time $t = 2$, which means the decision maker does not know which of the scenarios will happen at this stage. The same argument holds for each bundle. Hence, let say, the second decion for the bundle A_1 is $\boldsymbol{\beta}$ and the decision for the bundle A_2 is $\boldsymbol{\gamma}$. We can write the policy as:

$$\mathbf{X}(s) = \begin{cases} (\boldsymbol{\alpha}, \boldsymbol{\beta}, \cdot, \cdot) & s \in \{s^1, s^2, s^3\} \\ (\boldsymbol{\alpha}, \boldsymbol{\gamma}, \cdot, \cdot) & s \in \{s^4, s^5, s^6, s^7\} \end{cases}$$

Using same arguments as previously, the decision maker end up in stage 3 with the following policy.

$$\mathbf{X}(s) = \begin{cases} (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}, \cdot) & s \in \{s^1, s^2\} \\ (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\epsilon}, \cdot) & s = s^3 \\ (\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\eta}, \cdot) & s = s^4 \\ (\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\theta}, \cdot) & s \in \{s^5, s^6, s^7\} \end{cases}$$

Here we can say, that for the choice of decisions $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\epsilon}, \cdot)$ at this time period, whatever decision will happen at stage 4, we will end up with scenario s^3.

$$\mathbf{X}(s) = \begin{cases} (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}, \boldsymbol{\iota}) & s = s^1 \\ (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}, \boldsymbol{\kappa}) & s = s^2 \\ (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\epsilon}, \boldsymbol{\lambda}) & s = s^3 \\ (\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\eta}, \boldsymbol{\mu}) & s = s^4 \\ (\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\theta}, \boldsymbol{\nu}) & s = s^5 \\ (\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\theta}, \boldsymbol{\xi}) & s = s^6 \\ (\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\theta}, \boldsymbol{\pi}) & s = s^7 \end{cases}$$

After the last stage ($t = 4$), each scenario is uniquely determined by a collection of decisions, the policy $\mathbf{X}(s)$, at each time period. It is easily seen that the graph or the table 3.1.1 has the root-leaf structure. The selected scenario represents a path from the root (the first decision $\mathbf{x}_1(s)$) to the leaf (the last decision $\mathbf{x}_4(s)$). In general, we can suppose that the partition $\mathcal{A}_{t+1}$ is a refinement of the partition $\mathcal{A}_t$.

We will denote the space of all mappings $\mathbf{X} : S \to \mathbf{R}^n$ by $\mathcal{E}$. The polices $\mathbf{X}$ in the subspace

$$\mathcal{N} := \{\mathbf{X} \in \mathcal{E} : \mathbf{x}_t(s) \text{ is constant on each bundle } A \in \mathcal{A}_t \text{ for } t = 1, ..., T\} \tag{3.2}$$

are termed to be *implementable*. The policies belonging to the set

$$\mathcal{C} := \{\mathbf{X} \in \mathcal{E} : \mathbf{X}_t(s) \in C_s \quad \forall s \in S\} \tag{3.3}$$

are called *admissible*. The set C_s denotes the feasible set. The policy which is both admissible and implementable is what we shall mean by a *feasible* policy. To obtain an implentable policy we can assign to each scenario $s \in S$ a *weight* p_s that reflects its relative importance. The weight p_s has to satisfy the following conditions:

$$p_s > 0 \quad \forall s \in S, \text{ and } \sum_{s \in S} p_s = 1$$

Using weights we can calculate for all t and for every $A \in \mathcal{A}(t)$ the vector

$$\mathbf{X}_t(A) := \frac{\sum_{s \in A} p_s \mathbf{X}_t(s)}{\sum_{s \in A} p_s}. \tag{3.4}$$

Then we can define a new policy

$$\hat{\mathbf{X}}_t(s) = \mathbf{X}_t(A) \quad \forall s \in A. \tag{3.5}$$

Because $\mathbf{X}_t(A)$ is implementable so it is also the $\hat{\mathbf{X}}_t(s)$. The transformation

$$J : \mathcal{E} \ni \mathbf{X} \mapsto \hat{\mathbf{X}} \in \mathcal{N} \text{ defined by (3.4) - (3.5)} \tag{3.6}$$

represents an *aggregation operator*. The aggregation operator is a linear transoformation and it satisfies $J^2 = J$. For further information about the scenario aggregation see [9].

3.2. PHA for multi-stage SP

In the previous section we took closer look at the scenario aggregation which makes the policy $\mathbf{X}$ implementable. This will be the usefull tool to construct the progressive hedging algorithm. Introducing the weights p_s at the outset we have the following multi-stage SP derived from (3.1):

$$\begin{array}{ll} \min & \sum_{s\in S} p_s f_s(\mathbf{X}(s)) \\ \text{s.t.} & \mathbf{X} \in \mathcal{C} \cap \mathcal{N}, \end{array} \tag{3.7}$$

Our aim is to find a well hedgeg decision policy $\mathbf{X}$. The optimal solution $\mathbf{X}^* \in \mathcal{C} \cap \mathcal{N}$ is implementable and admissible, hence it is feasible. However simple the model (3.7) looks like, the problem is much larger and therefore harder to solve than solving the individual scenario subproblems $(\mathcal{P}_s)$ in (3.1).

To solve such program the *progressive hedging algorithm* is used. Algorithm consist of approximation of $\mathbf{X}^*$ by $\hat{\mathbf{X}}^0$ which is the solution of modified scenario subproblem $(\mathcal{P}_s^0)$ by "orientation prices" and penalties. At iteration i we take a contingent policy $\mathbf{X}^i$ obtained by solving the modified scenario subproblem $(\mathcal{P}_s^i)$ and aggregate it into an implementable policy $\hat{\mathbf{X}}^i$. Repeating this process we end up with a sequence of polices $\hat{\mathbf{X}}^i$ converging to the solution $\mathbf{X}^*$. The advantage of this algorithm is that we can stop at some point of the procedure and have a solution estimate better than $\hat{\mathbf{X}}^0$ or any other estimate until now. Lets look at the formal structure (see [9, 10]).

General formulation

Interpret the weights p_s as probabilities. *Expectations* $E(\mathbf{X}(s))$ are computed as:

$$E(\mathbf{X}(s)) = \sum_{s\in S} p_s \mathbf{X}(s)$$

The *conditional expectations* are $\mathbf{X}_t(A) = E(\mathbf{X}(s)|A)$, where $s \in A$. $\mathcal{E}$ represents an Euclidean space with the Euclidean norm $|\cdot|$ on $\mathbf{R}^n$ defined as

$$\|\mathbf{X}\| = [E(|\mathbf{X}(s)|)^2]^{1/2}$$

Then the aggregation operator J is the *orthogonal projection* on the subspace $\mathcal{N}$. Define new operator K as

$$K = I - J \qquad (K\mathbf{X} = \mathbf{X} - \hat{\mathbf{X}})$$

Using the terms above we can rephrase the problem (3.7) as:

$$\begin{array}{ll} \min & E(f_s(\mathbf{X}(s))) \\ \text{s.t.} & \mathbf{X} \in \mathcal{C}, K\mathbf{X} = 0 \end{array} \tag{3.8}$$

and then calculate for any $\mathbf{X}$ the corresponding $\hat{\mathbf{X}} = J\mathbf{X}$ and hence $\mathbf{X} - \hat{\mathbf{X}} = K\mathbf{X}$

Progessive Hedging Algorithm for Multi-stage Stochastic Programming

Step 1: Initialization: Choose the *penalty parameter* $r > 0$ and the *termination parameter* $\epsilon > 0$. Set the iteration $i = 1$, the *price system* $\mathbf{W}^0(s) = \mathbf{0}$ and the policy $\hat{\mathbf{X}}^0(s) = \mathbf{0}$ $\forall t$ and $\forall s \in S$.

Step 2: Solve the scenario subproblems $(\mathcal{P}_s^i)$ for all $s \in S$

$$\begin{aligned} \min \quad & f_s(\mathbf{x}_1(s), ..., \mathbf{x}_T(s)) + \sum_{t=1}^{T} \left(\mathbf{x}_t^T(s) \cdot \mathbf{w}^{i-1}(s) + \frac{r}{2} \parallel \mathbf{x}_t(s) - \hat{\mathbf{x}}_t^{i-1}(s) \parallel^2 \right) \quad (3.9) \\ \text{s.t.} \quad & \mathbf{x}_1(s), ..., \mathbf{x}_T(s) \in \mathcal{C}_s. \end{aligned}$$

Write down the scenario solution policy $\mathbf{X}^i(s) = (\mathbf{x}_1^i(s), ..., \mathbf{x}_T^i(s))$ $\forall s \in S$.

Step 3: Aggregate the scenario solution policy $\mathbf{X}^i(s)$ to $\hat{\mathbf{X}}^i(s)$ using equations (3.4) and (3.5), i.e.:

$$\hat{\mathbf{X}}^i(s) := \frac{\sum_{s \in A} p_s \mathbf{X}^i(s)}{\sum_{s \in A} p_s}.$$

Step 4: Compute $\pi^i := \sum_{s \in S} p_s \| \mathbf{X}^i(s) - \hat{\mathbf{X}}^i(s) \|$

Step 5: Check if the termination condition holds: $\pi^i < \epsilon$.

If yes, then stop. The policy $\hat{\mathbf{X}}^i(s) = (\hat{\mathbf{x}}_1^i(s), ..., \hat{\mathbf{x}}_T^i(s))$ is the solution of (3.7).

Else update the price system $\mathbf{w}_t^i(s) := \mathbf{w}_t^{i-1}(s) + r(\mathbf{x}_t^i(s) - \hat{\mathbf{x}}_t^i(s))$, set $i = i + 1$ and return to the step 2.

The distance term $\| \mathbf{X}^i(s) - \hat{\mathbf{X}}^i(s) \|$ can be interpreted as a measure of how far we are from satisfying all the constraints. We force this distance to zero with the penalty part of the objective function in (3.9) which has the form $\frac{r}{2} \parallel \mathbf{x}_t^i(s) - \hat{\mathbf{x}}_t^{i-1}(s) \parallel^2$. Choosing the value of the penalty parameter r we influence the progress of how our approximated policy $\hat{\mathbf{X}}(s)$ tends to the optimal solution $\mathbf{X}^*$. For further information about PHA MS SP see [11, 12, 13].

The Chapter 4 will be focused on two-stage problems and for this purpose, we will derive the PHA for two stage problems.

Progessive Hedging Algorithm for Two-stage Stochastic Programming

Before the algorithm will be presented, there are few things to be considered. We are trying to minimize the value of the objective function, which consists of the first-stage decision variable $\mathbf{x}$ and the second-stage decision variable $\mathbf{y}(s)$, written in a policy $\mathbf{X} = (\mathbf{x}, \mathbf{y}(s))$.
The problem is described as (2.12). If we initialize the price system to zero, $\mathbf{W}^0(s) = (\mathbf{w}_x^0, \mathbf{w}_y^0) = \mathbf{0}$, the second component $\mathbf{w}_y$ will remain 0 $\forall s \in S$ and $\forall i$. After the second-stage decision $\mathbf{y}(s)$ we end up with uniquely determined bundles. This will simplify the aggregation of the $\hat{\mathbf{y}}(s)$ in (3.5).

$$\hat{\mathbf{y}}(s) = \frac{\sum_{s \in A} p_s \mathbf{y}(s)}{\sum_{s \in A} p_s} = \mathbf{y}(s)$$

Hence the termination term π^i will look like

$$\pi^i = \sum_{s \in S} p_s \|\mathbf{X}^i(s) - \hat{\mathbf{X}}^i(s)\| = \sum_{s \in S} p_s \|\mathbf{x}^i(s) - \hat{\mathbf{x}}^i\|$$

The aggregated policy $\hat{\mathbf{X}} = (\hat{\mathbf{x}}, \hat{\mathbf{y}}(s))$ is the result of (3.7). The algorithm for the two-stage progressive hedging has the following form.

Step 1: Initialization: Choose $r \geq 0$ and $\epsilon \geq 0$. Set $i = 1$, $\mathbf{w}^0(s) = 0$ and $\hat{\mathbf{x}}^0 = 0$ $\forall t$ and $\forall s \in S$

Step 2: Solve scenario subproblems $(\mathcal{P}_s^i)$ all $s \in S$.

$$\begin{aligned} \min \quad & f(\mathbf{x}(s), \mathbf{y}(s)) + \mathbf{x}^T(s)\mathbf{w}^{i-1}(s) + \frac{r}{2} \parallel \mathbf{x}(s) - \hat{\mathbf{x}}^{i-1} \parallel^2 \\ \text{s.t.} \quad & \mathbf{x}(s), \mathbf{y}(s) \in \mathcal{C}_s. \end{aligned} \tag{3.10}$$

By solving the IS (3.10) we obtain the first-stage decison $\mathbf{x}^i(s)$ and the second-stage decision $\mathbf{y}^i(s)$ $\forall s \in S$ at the iteration i.

Step 3: Aggregate the IS solutions to $\hat{\mathbf{x}}^i$ and $\hat{\mathbf{y}}^i$.

$$\begin{aligned} \hat{\mathbf{x}}^i &:= \sum_{s \in S} p_s \mathbf{x}^i(s), \\ \hat{\mathbf{y}}^i(s) &:= \mathbf{y}^i(s) \quad \forall s \in S. \end{aligned}$$

Step 4: Compute $\pi^i := \sum_{s \in S} p_s \|\mathbf{x}^i(s) - \hat{\mathbf{x}}^i\|$

Step 5: Check if the termination condition holds: $\pi^i < \epsilon$.

If yes, then stop. The aggregated $\hat{\mathbf{x}}^i$ and $\hat{\mathbf{y}}^i(s)$ are solutions of (3.7).

Else update the price system $\mathbf{w}^i(s) := \mathbf{w}^{i-1}(s) + r(\mathbf{x}^i(s) - \hat{\mathbf{x}}^i)$, set $i = i + 1$ and return to the step 2.

For further information see [10], [14].

4. Real-Data Applications

4.1. Large-scale Problem

I start with the motivating example of the real-world problem. During the winter semester I was asked by Ing. Radovan Šomplák, Ph.D. to participate on the following application of WS stochastic problem:

Consider the network flow problem consisting of one type of waste and three methods how to deal with it: waste utitlization, disposal and other usage. Each city keeps the evidence of the number of waste disposal, waste utilization and other usage of a waste. The problem is that the production of a waste at some cities does not equal to three mentioned methods for those cities. Our aim is to find the network flow of a waste and to compare disposal and waste utilization using the probability weight. For our purpose we have data from municipalities with extended powers. Each municipality knows the production of waste, waste utilization and disposal for the time period of 7 years.

All the indices, parameters and variables used in our model are shown below.

Indices			*Parameters*	
i	city;	$i = 1, ..., 206$	$d_{i,j}$	distance between cities i and j
j	city;	$j = 1, ..., 206$	$p_{j,k}$	waste production in city j and year k,
k	year;	$k = 2009, ..., 2015$	$v_{i,k}$	waste utitlization in city i and year k,
s	simulation;	$s = 1, ..., 100$	$o_{i,k}$	waste disposal in city i and year k,
			$A_{i,j}$	element of adjacency matrix
			α_i^s	probability weight for city i, $\alpha_i^s \in (0, 1)$.

Variables

$x_{j,i,k}^s$ number of transported units of waste for waste utilization from city i to j in year k,

$y_{j,i,k}^s$ number of transported units of waste for disposal from city i to j in year k.

Instead of using the word "scenario" as set of indices s, I am using the word "simulation". This is done because we miss the typical scenario-tree structure. Changing the probability weight α_i^s at each simulation can be interpreted as WS approach, because we know α_i^s before the decision is made. The problem (4.1) is linear, one-stage, WS problem and has the form: $\forall s$ solve the problem

$$\begin{aligned}
\min \quad & \sum_j \sum_i \sum_k \alpha_i d_{i,j} x_{j,i,k} + \sum_j \sum_i \sum_k (1 - \alpha_i) d_{i,j} y_{j,i,k} && && (4.1)\\
\text{s.t.} \quad & \sum_j A_{i,j} x_{j,i,k} = v_{i,k} && \forall i, k,\\
& \sum_j A_{i,j} y_{j,i,k} = o_{i,k} && \forall i, k,\\
& \sum_i A_{i,j} (x_{j,i,k} + y_{j,i,k}) \leq p_{j,k} && \forall j, k,\\
& x_{j,i,k} \geq 0 && \forall i, j, k,\\
& y_{j,i,k} \geq 0 && \forall i, j, k.
\end{aligned}$$

The objective function minimizes the transportation costs using the probability weight α_i^s, which is computed as follows. Let us define auxiliary parameters $\beta_i \in (0,1)$ and γ_i.

$$\gamma_i(\beta_i) := \begin{cases} 4\beta_i, & \text{if } 0 \leq \beta_i < 0.5, \\ 4 - 4\beta_i, & \text{if } 0.5 \leq \beta_i \leq 1. \end{cases}$$

$$\alpha_i^s := \beta_i \gamma_i$$

This improvement will give less weight to less likely situations and, on the contrary, add more weight to more likely situations.

Next two constraints, $\sum_j A_{i,j} x_{j,i,k} = v_{i,k} \quad \forall i,k$ and $\sum_j A_{i,j} y_{j,i,k} = o_{i,k} \quad \forall i,k$, represent the restriction to the number of transported waste.
The constraint $\sum_i A_{i,j}(x_{j,i,k} + y_{j,i,k}) \leq p_{j,k} \quad \forall j,k$ is the action against overflowing the production.

If we consider one hundred simulations of (4.1) we may label the problem as "what if" analysis. As a result we obtain thousands of values changing in the probability, in the each city throughout years. There may arise a question, what are the results? Rather than values, I'll show some of the graphical results. All the data were processed in MS Excel. In the figure 4.1.1 are showed the results for the disposal of the capital city Prague. This figure shows the simulation as a more possible values in each year.

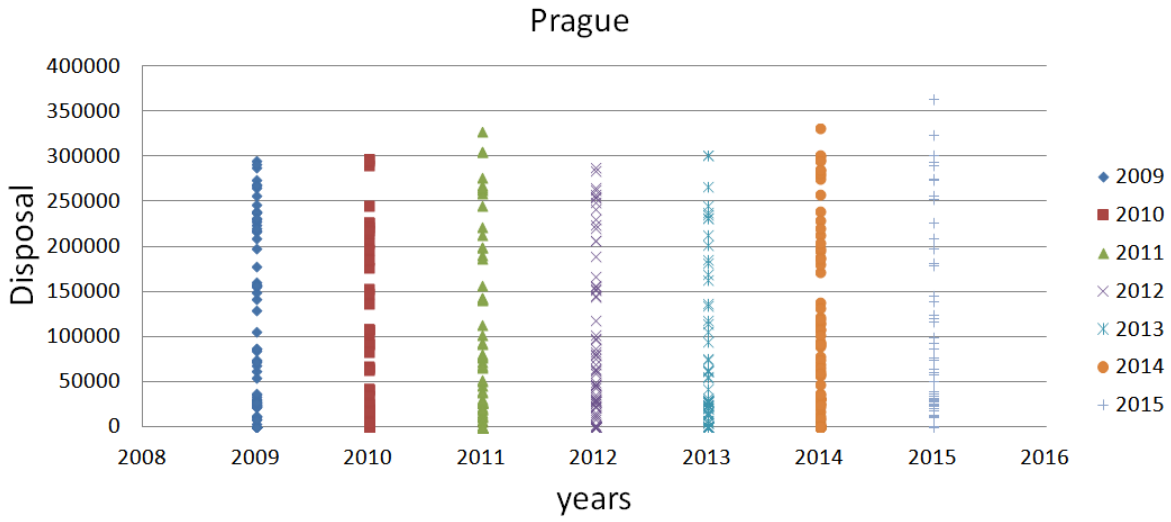

Figure 4.1.1: Simulations with different probability weight α_i^s throughout the years.

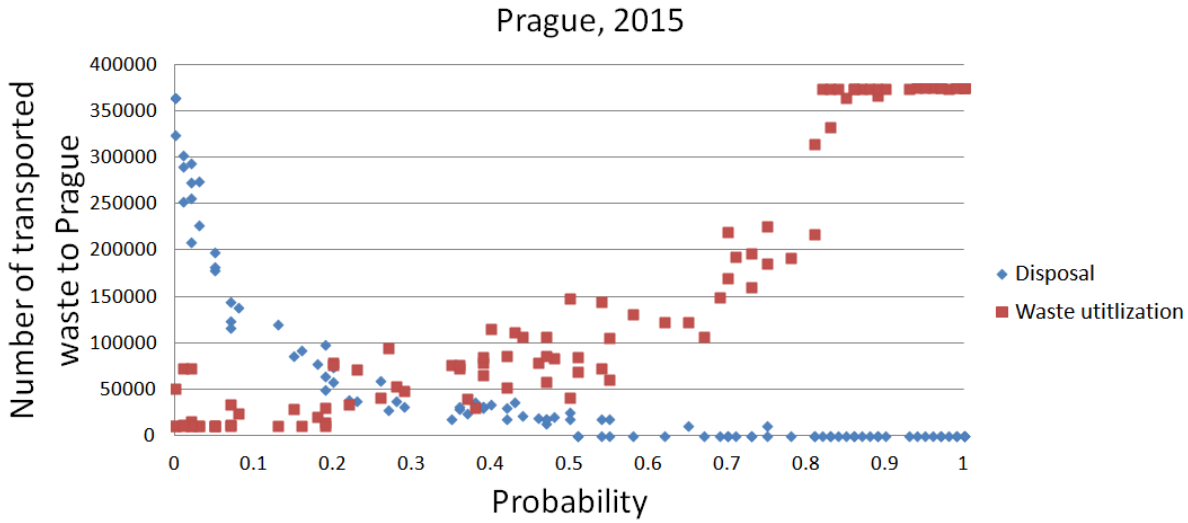

Figure 4.1.2: Dependence of waste utilization $x^s_{i,j,k}$ and disposal $y^s_{i,j,k}$ on prob. weight α^s_i.

In the figure 4.1.2 is the dependency of waste utitlization $x^s_{i,j,k}$ and disposal $y^s_{i,j,k}$ on probability α^s_i for the capital city Prague during the year 2015. The figure represents the number of units of waste transported to Prague to be disposed or utilized. Both curves have exponential behaviour given by $\alpha^s_i x^s_{i,j,k}$ for waste utilization and $(1-\alpha^s_i)y^s_{i,j,k}$ for disposal from the objective function of (4.1).

Previous two figures are related to the one city, but we can use filters in MS Excel in a slightly different way and do some case study for the whole region. Remaining two figures, 4.1.3 and 4.1.4, are related to the South Moravian region.

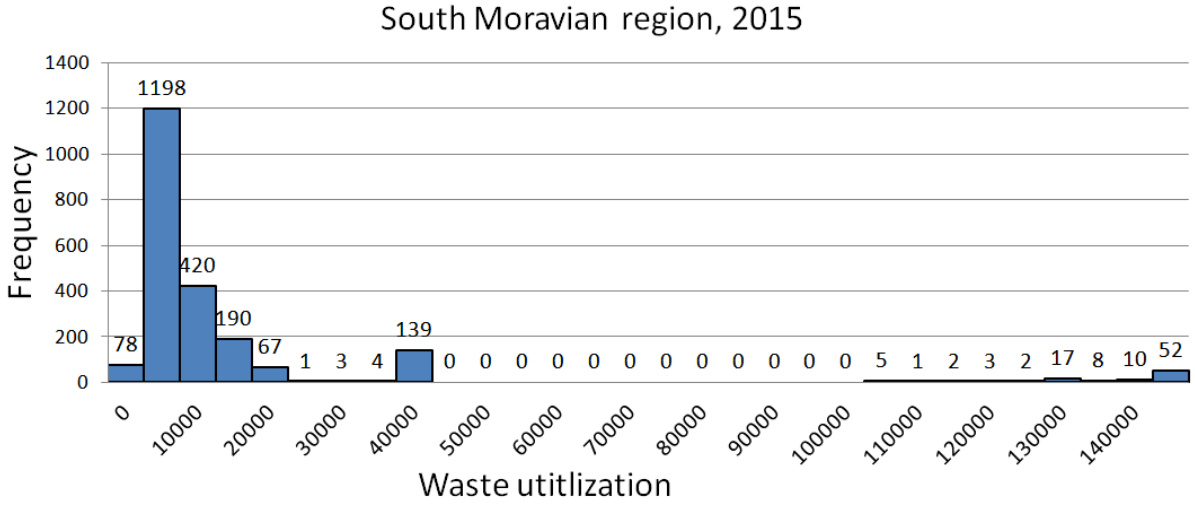

Figure 4.1.3: Histogram.

In the figure 4.1.3 is the histogram of waste utilization $x^s_{i,j,k}$ for the South Moravian region during the year 2015. Intervals are taken over 5000 units of waste. Values over 100000 belogs to the city Brno for all simulations. The last figure, 4.1.4, shows the huge difference in transported waste between the waste utitlization $x^s_{i,j,k}$, disposal $y^s_{i,j,k}$ and the other waste processing, where the other waste processing is computed as:

$$m_{j,k} = p_{j,k} - (\sum_i y_{j,i,k} + x_{j,i,k}).$$

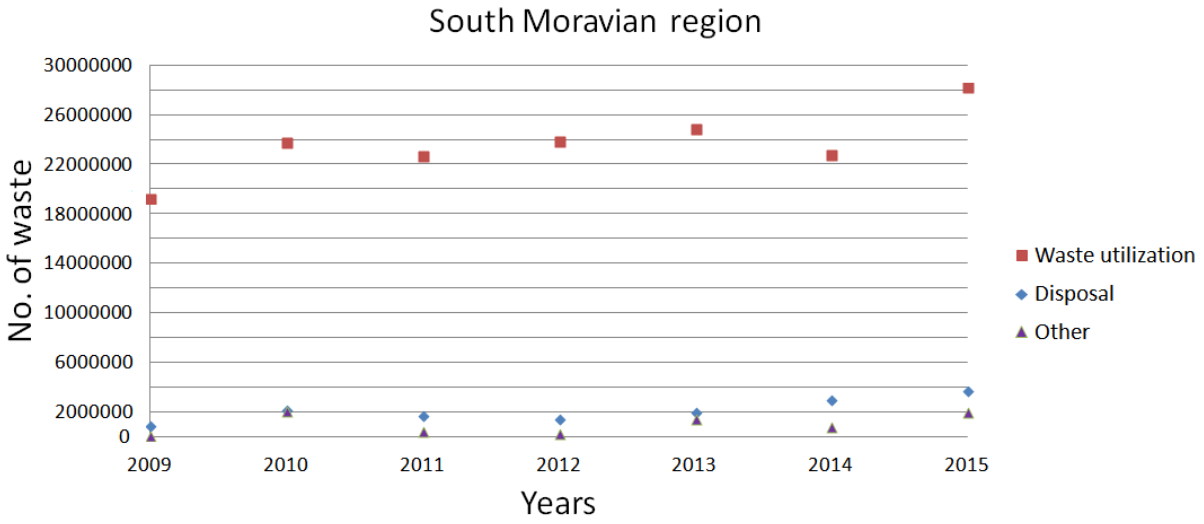

Figure 4.1.4: Total amount of transported waste for the South Moravian region throughout the years.

4.2. Farmer's Problem

To apply PHA we begin with a model, which is detaily described in J.R. Birge's Introduction to stochastic programming, see [8]. Our ideal model is represented as a Farmer's Problem. Secondly, because the final results are available in the book, we can reformulate the problem and use the progressive hedging algorithm, (3.9), to solve it numericaly.

"Consider an European farmer who specializes in raising grain, corn, and sugar beets on his 500 acres of land. During the winter, he wants to decide how much land to devote to each crop. After the season farmer wants to maximize his profit and still meet the minimum requirements to feed the cattle." [8]

All known parameters are described in the table below:

Parameter	Description	Wheat	Corn	Sugar Beets
$e_{s,i}$	Yield (T/acre)	2/2.5/3	2.4/3/3.6	16/20/24
c_i, $i = 1, 2, 3$	Planting cost (\$/acre)	150	230	260
d_j, $j = 1, 2, 3$	Selling price (\$/T)	170	150	36 under g = 6000 T
d_j, $j = 4$				10 above 6000 T
d_j, $j = 5, 6$	Purchase price (\$/T)	238	210	–
f_i	Min. requirement (T)	200	240	–
b	Total available land: 500 acres			

Scenarios	Description		Probability P_s
s_1	Below average yields	(-20%)	1/3
s_2	Expected yields	(+0%)	1/3
s_3	Above average yields	(+20%)	1/3

Let us denote the following variables:

Variables

x_1	wheat (acres),	$w_{1,s}$	wheat sold (T),
x_2	corn (acres),	$y_{1,s}$	wheat purchased (T),
x_3	sugar beets (acres),	$w_{2,s}$	corn sold (T),
		$y_{2,s}$	corn purchased (T),
		$w_{3,s}$	sugar beets sold - favorable price (T),
		$w_{4,s}$	sugar beets sold - lower price (T).

For our purpose it will be convenient to split the first-stage decisions and the second-stage decisions.

$x_i = (x_1, x_2, x_3)$	first-stage decision
$y_{j,s} = (w_{1,s}, w_{2,s}, w_{3,s}, w_{4,s}, y_{1,s}, y_{2,s})$	second-stage decision

As was described in the introduction to the problem, during the first-stage farmer decides, how much land will be devoted to the wheat, corn or sugar beets. The second-stage decision is taken after the season, during the harvest, when the farmer decides how much of crop to sell, to buy or to keep for his cattle. The first-stage decisios are according to the HN approach and the second-stage decisions are taken as the WS approach.

The model has the following form:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{3} c_i x_i + \sum_{s=1}^{3} P_s \Big(\sum_{j=1}^{6} d_j y_{j,s}\Big), && && (4.2)\\
\text{s.t.} \quad & \sum_{i=1}^{3} x_i \le b, && \\
& e_{s,1}x_1 + y_{5,s} - y_{1,s} \ge f_1 && \forall s,\\
& e_{s,2}x_2 + y_{6,s} - y_{2,s} \ge f_2 && \forall s,\\
& y_{3,s} + y_{4,s} \le e_{s,3}x_3 && \forall s,\\
& y_{3,s} \le g && \forall s,\\
& x_i, y_{j,s} \ge 0 && \forall i, j, s.
\end{aligned}$$

If we want to reformulate the model (4.2) into the scneario subproblems (3.9) used for PHA, the only change will be inside the objective function. Adding the price-term and the penalty-term we obtain the model (4.3).

$$\begin{aligned}
\min \quad & \sum_{i=1}^{3} c_i x_i + \sum_{j=1}^{6} d_j y_{j,s} + \sum_{i=1}^{3} \Big(x_i w_i + \frac{r}{2} \parallel x_i - \hat{x}_i \parallel^2\Big) \quad \forall s, && && (4.3)\\
\text{s.t.} \quad & \sum_{i=1}^{3} x_i \le b, && \\
& e_{s,1}x_1 + y_{5,s} - y_{1,s} \ge f_1 && \forall s,\\
& e_{s,2}x_2 + y_{6,s} - y_{2,s} \ge f_2 && \forall s,\\
& y_{3,s} + y_{4,s} \le e_{s,3}x_3 && \forall s,\\
& y_{3,s} \le g && \forall s,\\
& x_i, y_{j,s} \ge 0 && \forall i, j, s.
\end{aligned}$$

Setting the initial price system $w_i^0 = 0$ and the initial policy $\hat{x}_i^0 = 0$ we can iteratively solve the initial problem (4.2) with predefined desired accuracy ϵ and penalty parameter r. Results for the first and the second-stage are shown in tables 4.1 and 4.2.

$\epsilon = 0.001$, $r = 1$			
i	x_1	x_2	x_3
1	129.111	81.8333	289.056
2	123.704	86.8333	289.463
3	121.139	96.7191	282.142
4	124.287	99.6718	276.041
5	128.718	97.5315	273.751
6	133.431	95.3738	271.195
7	141.294	96.2366	262.47
8	149.781	97.7232	252.496
9	157.021	97.9636	245.015
10	162.184	96.1266	241.689
⋮	⋮	⋮	⋮
107	170	79.9998	250
108	170	79.9999	250
	170	80	250

Table 4.1: First-stage decisions.

$\epsilon = 0.001$, $r = 1$						
i	$w_{1,s}$	$w_{2,s}$	$w_{3,s}$	$w_{4,s}$	$y_{1,s}$	$y_{2,s}$
108	140.001	0	4000	0	0	48.0004
	225.001	0	4999.99	0	0	0
	310.001	47.9991	6000	0	0	0
	140	0	4000	0	0	48
	225	0	5000	0	0	0
	310	48	6000	0	0	0

Table 4.2: Second-stage decisions.

For the chosen $\epsilon = 0.001$ and $r = 1$ the PHA needs 108 iterations i. The last row in the table 4.1 without the number of iteration is the exact result of problem (4.2) and similarly, the part of the table 4.2, which is below, contains the result for (4.2). If we repeat the computation with the same ϵ but the r is changed, it is easily seen the dependence of the chosen r on the number of iterations and the behavior of the convergence in the figure 4.2.1

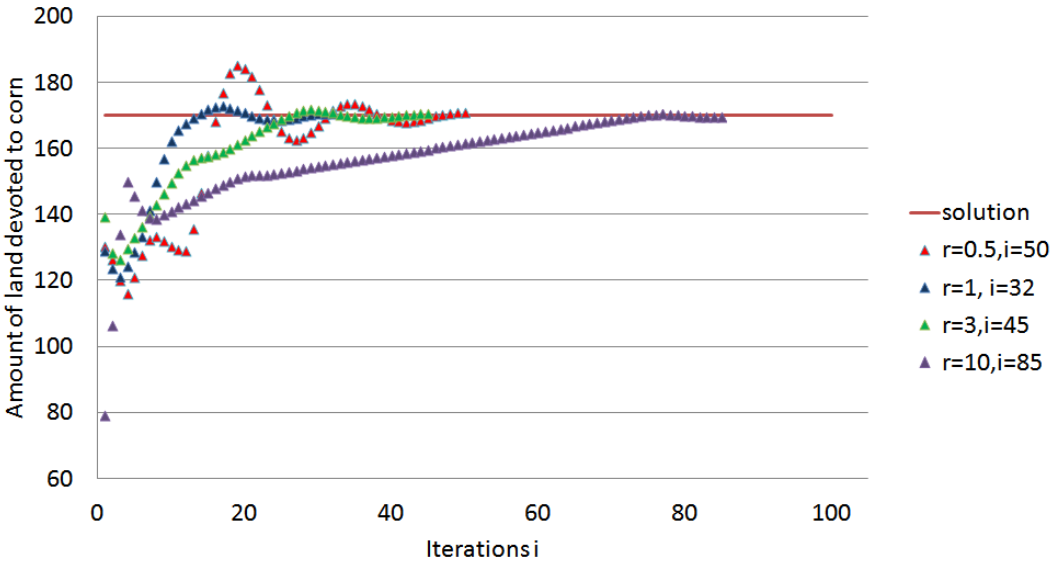

Figure 4.2.1: Changes with different penalty parameter.

4.3. Main Problem

This section can be considered as a sequel of my bachelor's thesis, see [4]. Before I test real-world data, I will consider the following problem with the smaller data.

4.3.1. Smaller Data

Let us consider the flow network showed in the figure 4.3.1 consisting of cities (blue square), traffic points (black dot), landfills (magenta circles), possible places for incinerators (green circles) and arcs between the nodes.

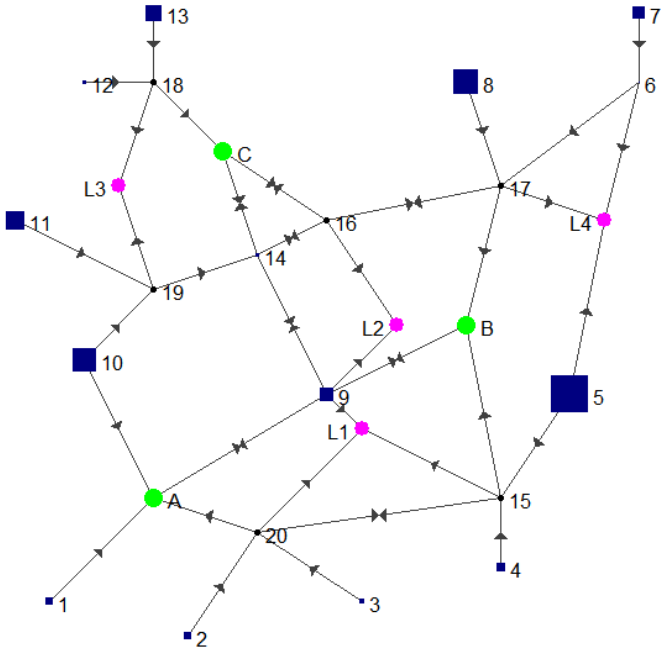

Figure 4.3.1: Network for testing smaller data.

Our aim is to build at least one incinerator and see, if our investment came back or not. We want to maximize the profit from landfills and incinerators by collecting the waste in cities. Also, there is the possibility to pay the penalty instead of collecting the waste.

We are dealing with the two-stage stochastic problem, in which the first decision will be to build or not the incinerator/incinerators and during the second-stage decision, which will be made after couple of years, we will transport as much waste as possible to obtain the optimal solution. All the indices, parameters and variables are listed below:

Indices

$i \in N$	index of nodes	$i = 1, ..., 20, A, B, C, L_1, ..., L_4$
$k \in K$	index of coordinates	$k = Xcoor, Ycoor$
$e \in A$	index of arcs	$e = 1 - A, 2 - 20, ..., 6 - L_4$
$s \in S$	index of scenarios	$s = s_1, s_2, s_3$
$ic \in N_c \subset N$	index of cities	$ic = 1, ..., 14$
$it \in N_t \subset N$	index of traffic points	$it = 15, ..., 20$
$is \in N_s \subset N$	index of incinerators	$is = A, B, C$
$il \in N_l \subset N$	index of landfills	$il = L_1, L_2, L_3, L_4$

Parameters and scalars	
$n_{i,k}$	coordinates of nodes
$M_{i,e}$	element of incidence matrix
c_e	transportation costs
g_s	profit in the incinerator
q_s	penalty for waste not removed
$t_{ic,s}$	quantity of trash in cities
d_e	distance by arcs
Pr_s	probability of scenario
h_s	profit in landfills
f	freight in coins per km using lorry
b	cost to build the incinerator
α	upper bound for penalty

Variables	
$y_{e,s}$	amount of transported units
z	total profit
$p_{ic,s}$	number of penalty units
x_{is}	build or not incinerators
$k_{is,s}$	capacity of the incinerator

Before the computation of model (4.4), we may decompose an arc $e \in A$ to its incident nodes $i \in N$ and $j \in N$ and use this to compute the euclidean distance between nodes:

$$d_e = d_{i,j} = \sqrt{\sum_k (n_{i,k} - n_{j,k})}$$

The transportation cost is then expressed as $c_e = f \cdot d_e$. We can set equal probability to each scenario:

$$Pr_s = \frac{1}{\mid S \mid} \; \forall s \in S, \text{ satisfying } \sum_s Pr_s = 1.$$

The production of waste in each city $t_{ic,s}$ differs according to the scenario s. In our case, we admit 3 scenarios, each with probability $Pr_s = 1/3 \; \forall s \in S$.

$$t_{ic,s} = \begin{cases} +0\%, & s = s_1, \\ -20\%, & s = s_2, \\ +20\%, & s = s_3. \end{cases}$$

The first-stage variable x_{is} is a binary variable represented as:

$$x_{is} = \begin{cases} 1, & \text{build the incinerator,} \\ 0, & \text{otherwise.} \end{cases}$$

Decisions z, $p_{ic,s}$, $k_{is,s}$ are dependent variables.

The problem (4.4) has the following properties: Two-stage mixed integer stochastic problem with the first decision x_{is} made by HN approach and the second decision $y_{e,s}$ made by WS approach.

$$\begin{aligned}
\max \quad & \Big\{ -\sum_{is} x_{is} b + \sum_{s} Pr_s \Big\{ \sum_{is} \sum_{e} -M_{is,e} y_{e,s} g_s - \sum_{e} c_e y_{e,s} + \\
& \qquad + \sum_{il} \sum_{e} -M_{il,e} y_{e,s} h_s - \sum_{ic} p_{ic,s} q_s \Big\} \Big\} \qquad (4.4) \\
\text{s.t.} \quad & \sum_{is} x_{is} \geq 1, \\
& \sum_{e} M_{ic,e} y_{e,s} \leq t_{ic,s} && \forall ic, s, \\
& \sum_{ic} p_{ic,s} \leq \alpha \sum_{ic} t_{ic,s} && \forall s, \\
& t_{ic,s} - \sum_{e} y_{e,s} M_{ic,e} = p_{ic,s} && \forall ic, s, \\
& \sum_{e} M_{it,e} y_{e,s} = 0 && \forall it, s, \\
& (\sum_{ic} t_{ic,s}) x_{is} = k_{is,s} && \forall is, s, \\
& -(\sum_{e} M_{il,e} y_{e,s}) \leq k_{is,s} && \forall is, s, \\
& x_{is} \in \{0, 1\} && \forall is, \\
& y_{e,s} \geq 0 && \forall e, s, \\
& p_{ic,s} \geq 0 && \forall ic, s, \\
& k_{il,s} \geq 0 && \forall il, s.
\end{aligned}$$

Let us have a look at terms in the objective function. The first term $-\sum_{is} x_{is} b$ contains the first-stage decision. The sum over incinerators represents costs for building incinerators. Next terms represent the second-stage decisions. Here, we consider yields in landfills and incinerators $\sum_{is} \sum_{e} -M_{is,e} y_{e,s} g_s + \sum_{il} \sum_{e} -M_{il,e} y_{e,s} h_s$, transportation costs $-\sum_{e} c_e y_{e,s}$ and the penalty term, if we do not collect the waste $-\sum_{ic} p_{ic,s} q_s$.

The first constraint $\sum_{is} x_{is} \geq 1$, which states that at least one incinerator will be built, belongs to the first-stage decision, whereas the rest of the constraints belong to the second-stage. The next constraint, $\sum_{e} M_{ic,e} y_{e,s} \leq t_{ic,s} \quad \forall ic, s$, states that the transported units of waste cannot exceed the production in each city and each scenario. $\sum_{ic} p_{ic,s} \leq \alpha \sum_{ic} t_{ic,s} \quad \forall s$ gives us the possibility not to collect all the waste, but we can pay the penalty instead up to $\alpha\%$ of the total waste in the each scenario. The penalty is computed as the difference between the waste production and the number of transported units, $p_{ic,s} = t_{ic,s} - \sum_{e} y_{e,s} M_{ic,e} \quad \forall ic, s$. In our network, 4.3.1, we consider also traffic points. Traffic points can be interpreted as the main junctions and are mathematically represented as the nodes, in which the inflow equals the outflow. In the model (4.4), such nodes are underlined by $\sum_{e} M_{it,e} y_{e,s} = 0 \quad \forall it, s$ setting that there is no production of waste. The dependent variable $k_{is,s}$ is computed as $k_{is,s} = (\sum_{ic} t_{ic,s}) x_{is} \quad \forall is, s$ and is used in the last constraint, $-(\sum_{e} M_{il,e} y_{e,s}) \leq k_{is,s} \quad \forall is, s$, where the inflow to incinerators cannot exceed the capacity of incinerators.

In flow networks 4.3.2 and 4.3.3 are two results, one for the situation with preferred yields in incinerators and the other one, where the yields are similar to yields in landfills. Both figures represent the first scenario. Flow networks for other scenarios look similar.

The following results for both networks are enlisted bellow:

Description			Results for 4.3.2	Results for4.3.3
Freight - lorry	f	[Kč/T/km]	2	4
Building costs	b	[Kč]	10 000	20 000
Profit - landfills	h_s	[Kč/T]	(50,80)	(90,110)
Profit - incinerators	g_s	[Kč/T]	(100,130)	(100,130)
Penalty - cost	q_s	[Kč]	(40,60)	(40,60)

First-stage decision			
Incinerators - built	x_{is}	A, B	A

Second-stage decisions									
			node	s_1	s_2	s_3	node	s_1	s_2	s_3
Penalty	$p_{ic,s}$	[T]	(All 0.000)				5	86.75	69.40	104.10
Capacity	$k_{is,s}$	[T]	A	1735	1388	2082	A	1735.0	1388.0	2082.0
			B	1735	1388	2082				
			edge	s_1	s_2	s_3	edge	s_1	s_2	s_3
Transported waste	$y_{e,s}$	[T]	1-A	64	51.2	76.8	1-A	64	51.2	76.8
			2-20	62	49.6	74.4	2-20	62	49.6	74.4
			20-A	110	88	132	20-A	110	88	132
			3-20	48	38.4	57.6	3-20	48	38.4	57.6
			4-15	73	58.4	87.6	4-15	73	58.4	87.6
			5-15	500	400	600	5-15	413.25	330.6	495.9
			15-B	573	458.4	687.6	10-A	199	159.2	238.8
			9-B	308	123.2	369.6	11-19	154	123.2	184.8
			10-A	199	159.2	238.8	8-17	200	160	240
			11-19	154	123.2	184.8	7-6	103	82.4	123.6
			17-B	360	288	432	12-18	30	24	36
			8-17	200	160	240	13-18	130	104	156
			7-6	103	82.4	123.6	14-9	36	28.8	43.2
			18-C	160	128	192	19-L3	154	123.2	184.8
			12-18	30	24	36	18-L3	160	128	192
			13-18	130	104	156	15-L1	486.25	389	583.5
			19-14	154		184.8	9-L1	154	123.2	184.8
			C-16	160	128	192	17-L4	200	160	240
			14-9	190	28.8	228	6-L4	121	96.8	145.2
			16-17	160	128	192				
			19-L3		123.2					
			6-L4	121	96.8	145.2				
Total profit	z	[Kč]	44 650.346				-49 243.901			

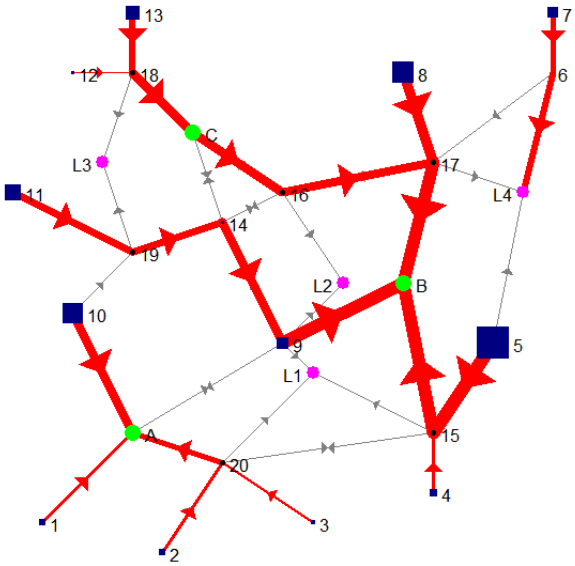

Figure 4.3.2: Network flow for the higher profits of incinerations.

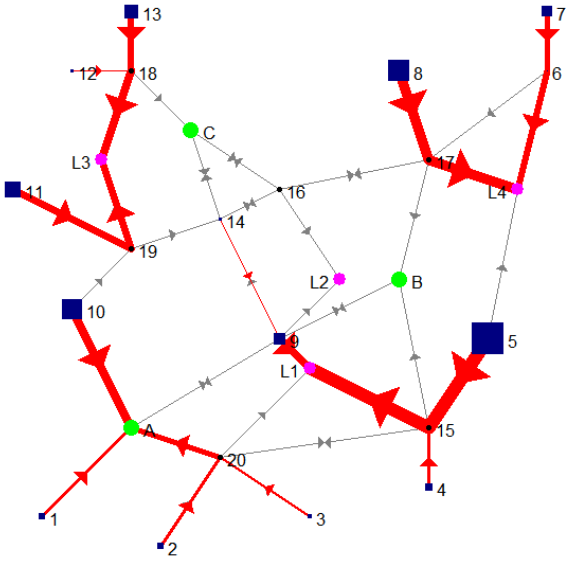

Figure 4.3.3: Network flow for the lower profits of incinerations.

4.3.2. Real-world Data

Now we can take a much bigger data, the real world data, for the model (4.4). For my purpose I have data of collection the municipal waste during the years 2009-2015. In this part, I will create a direct link to my bachelor's thesis, see [4], where I was looking for 10 potential optimal places for building incinerators considering all the original landfills and incinerators. Each city, landfill and incinerator have GPS coordinates. Potential coordinates of incinerators and regions are listed in the table below.

Node	X-GPS	Y-GPS	Region
1SPA	14.09	56.82	Středočeský
2SPA	17.60	56.22	Olomoucký
3SPA	15.77	56.84	Pardubický
4SPA	14.37	56.86	Hlavní město Praha
5SPA	16.60	55.92	Jihomoravský
6SPA	14.37	56.86	Hlavní město Praha
7SPA	15.13	56.57	Středočeský
8SPA	14.81	57.02	Středočeský
9SPA	14.37	56.86	Hlavní město Praha
10SPA	17.75	56.51	Moravskoslezský

In comparison with indices in the Section 4.3.1, we are considering municipalities with extended powers $ic \in N_c$, $|N_c| = 206$, already built landfills an incinerators $il \in N_l$, $|N_l| = 114$, set of potential incinerators $is \in N_s$, $|N_s| = 10$. The set of traffic points $it \in N_t$, $N_t = \emptyset$ is an empty set, because the set of edges $e \in A$, $|A| = 2069$, has sufficient cardinality for our purpose. This leads to the simplification of the model (4.4), in which the constraint $\sum_e M_{it,e} y_{e,s} = 0$ is redundant $\forall it, s$. The rest of the model (4.4) holds.

In the bachelor's thesis [4], the model was one-stage problem and all decisions were deterministic. Now the model is two-stage stochastic problem, in which the first-stage decision

x_i is a binary variable. It denotes whether the incinerator will be built or not. The second-stage decision $y_{e,s}$ is made after the period of seven years and it denotes the number of transported waste. As a result we obtain the network flow with the optimal number of built incinerators and their locations.

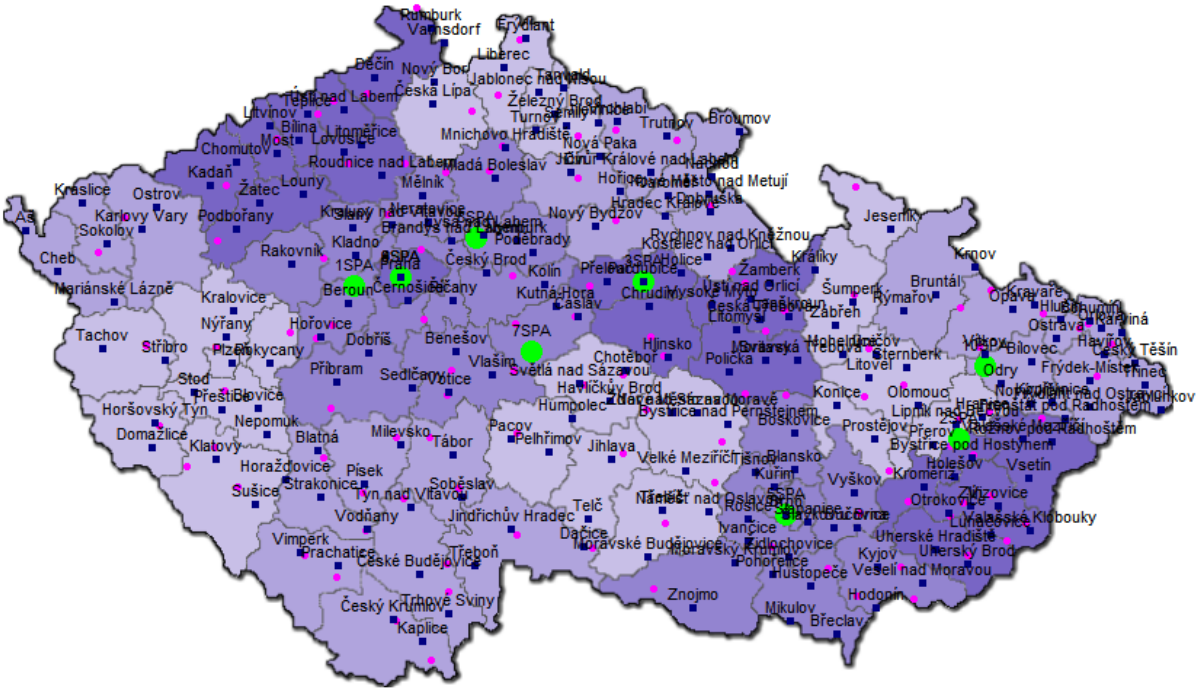

Figure 4.3.4: Layout of cities, landfills and possible incinerators.

The layout of cities (blue squares), landfills (magenta circles) and possible incinerators (green circles) can be seen in the figure 4.3.4. Regions are represented by different shade of violet. The waste distribution among regions in percentage is shown in the figure 4.3.5. Let us highlight the capital city Prague with the 13% of total waste comparing to regions.

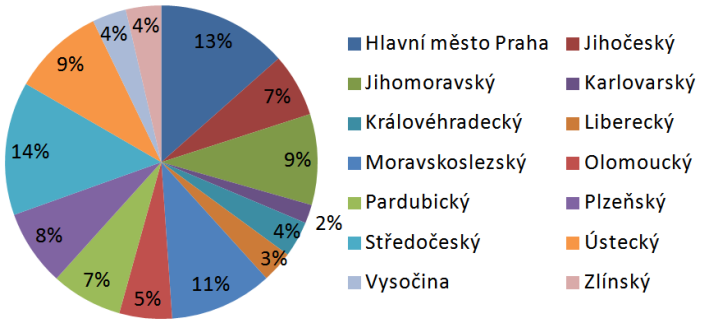

Figure 4.3.5: Waste distribution in regions.

4.3.3. Real-world Data using PHA

We can reformulate the model (4.4) into the PHA modification shown in (3.9) to obtain scenario subproblems. The model (4.5) is TS SMINLP[1].
$\forall s \in S$ solve the problem:

$$\begin{aligned}
\max \quad & \Big\{ -\sum_{is} x_{is} b + \sum_{is}\sum_{e} -M_{is,e} y_{e,s} g_s - \sum_{e} c_e y_{e,s} - \sum_{ic} p_{ic,s} q_s + \\
& + \sum_{il}\sum_{e} - M_{il,e} y_{e,s} h_s + \sum_{is} -(x_{is} w_{is,s} - \frac{r}{2}\|x_{is} - \hat{x}_{is}\|^2) \Big\} \qquad (4.5) \\
\text{s.t.} \quad & \sum_{is} x_{is} \geq 1, \\
& \sum_{e} M_{ic,e} y_{e,s} \leq t_{ic,s} && \forall ic, s, \\
& \sum_{ic} p_{ic,s} \leq \alpha \sum_{ic} t_{ic,s} && \forall s, \\
& t_{ic,s} - \sum_{e} y_{e,s} M_{ic,e} = p_{ic,s} && \forall ic, s, \\
& (\sum_{ic} t_{ic,s}) x_{is} = k_{is,s} && \forall is, s, \\
& -(\sum_{e} M_{il,e} y_{e,s}) \leq k_{is,s} && \forall is, s, \\
& x_{is} \in \{0,1\} && \forall is, \\
& y_{e,s} \geq 0 && \forall e, s, \\
& p_{ic,s} \geq 0 && \forall ic, s, \\
& k_{il,s} \geq 0 && \forall il, s.
\end{aligned}$$

Note, that now we are dealing with the maximization instead of the minimization as in (3.9). Hence the price term $-\sum_{is} x_{is} w_{is,s}$ and the penalization term $-\frac{r}{2}\sum_{is}\|x_{is} - \hat{x}_{is}\|^2$ have the minus sign to force the value of the objective function to the desired result.

In this step the difficulty arises. Imagine that we have a linear problem. Linear problems are convex. PHA changes the linear problem into the nonlinear one, but it preserves the convexity. However, if we have the TS SP, where some of the decisions have an integer character, we break the convexity. Such problem has the model (4.5), where the first-stage decision x_{is} is binary. For further information see [16].

Due to the nonconvnexity it is redundant to compute the termination parameter ϵ nor the stopping criteria π^i. The following algorithm shows how we can modify the classical PHA into the version for the TS SMIP. It can be found in [16].

[1]Two-stage stochastic mixed integer nonlinear program.

The Progressive Hedging Algorithm for TS SMIP

Step 1: Initialization: iteration $i = 0$, price term $w^i(s) = 0 \quad \forall s \in S$. For each $s \in S$ compute IS:

$$\begin{aligned} \max \quad & c^T x + g(s)^T y(s) \\ \text{s.t.} \quad & x, y(s) \in C(s) \end{aligned} \tag{4.6}$$

Step 2: Iteration update: $i = i + 1$

Step 3: Aggregation: $\hat{x}^i = \sum_{s \in S} P(s) x^i(s)$

Step 4: Price update: $w^i(s) = w^{i-1}(s) + r(x^i(s) - \hat{x}^i(s))$

Step 5: Decomposition: $\forall s \in S$ compute:

$$\begin{aligned} \max \quad & c^T x + g(s)^T y(s) - w^i(s)^T x - \frac{r}{2} \parallel x - \hat{x}^i \parallel^2 \\ \text{s.t.} \quad & x, y(s) \in C(s) \end{aligned} \tag{4.7}$$

Step 6: Termination: If all scenario solutions $x(s)$ are equal, stop. Else, go to step 2.

There is however one extra step in the initialization in form of (4.6) computed at $i = 0$. This step allows us to obtain classical IS solutions and update the price term $w^i(s)$ before the decomposition (4.7).

The value of the penalty parameter r will be much bigger than in the case of the Farmer's problem described in the Section 4.2. There the best setting was achieved for $r = 1$. We need to force the objective function z with a reasonably big value, not to be the major nor the minor part of z. To set the proper r experimentally, we can come up with new auxiliary parameters computing each part of the objective function in (4.7). For each IS we compute.

$$\begin{aligned} aux_1(s) &= c^T x + g(s)^T y(s) \\ aux_2(s) &= -w^i(s)^T x - \frac{r}{2} \parallel x - \hat{x}^i \parallel^2 \\ z &= aux_1(s) + aux_2(s) \end{aligned}$$

The bigger the r is, the bigger the force to change the value of the objective function. For next computations I choose $r = 10^8$.

Let us set parameters: $f = 3$, $b = 500,000,000$, $h_s \in (50; 80)$, $g_s \in (200; 250)$ and $q_s \in (10,000; 15,000)$. Using the model (4.4) we end up with the optimal solution. During the first-stage we build an incinerator at 1SPA. As you can see in the table 4.3.1 the PHA result set the value for 1SPA at 2/3 and also set 1/3 for 9SPA. Looking at the constraint $\sum_{is} x_{is} \geq 1$ in the model (4.5) we know, that at least one incinerator will be built. Considering the two non zero values, we will probably pick the higher one and hence choose the same decision in comparison with the model (4.4). The non zero value for 9SPA can be taken as a potential place to be considered. The GPS coordinates of 9SPA point to the capital city Prague. Values for PHA iterations $i = 0$, $i = 1$, $i = 5$ and the result using GAMS, *opt*, are shown in the figure 4.3.6. Incinerators 1SPA-10SPA are on the x-axis with the value of aggregated first-stage decision x_i on the y-axis. The null iteration represents aggregation of individual scenarios ($r = 0$).

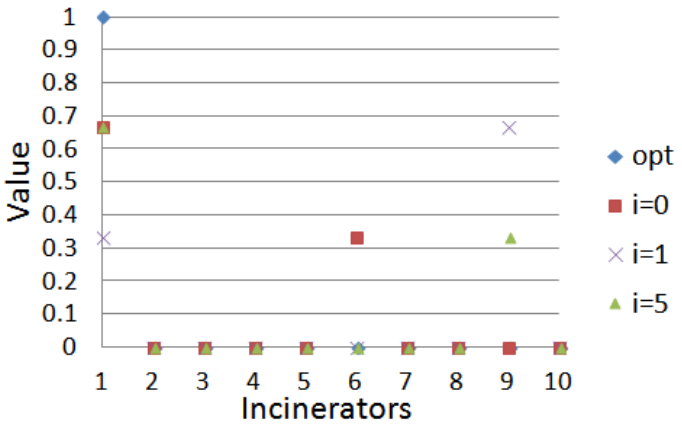

Figure 4.3.6: Comparison of the PHA result and the optimal result for the first-stage decision x_{is}.

	PHA			GAMS
	i=0	i=2	i=5	opt
1SPA	2/3	1/3	2/3	1
2SPA	0	0	0	0
3SPA	0	0	0	0
4SPA	0	0	0	0
5SPA	0	0	0	0
6SPA	1/3	0	0	0
7SPA	0	0	0	0
8SPA	0	0	0	0
9SPA	0	2/3	1/3	0
10SPA	0	0	0	0

Table 4.3.1: Results after the 5th iteration.

	z	r	i	solver
PHA	989,350,266.7	10^8	5	MINLP
GAMS	840,992,400			MIP

We end up with the bigger value of z using PHA, because of the aggregation principle. The value 2/3 at 1SPA means that the two of three IS set the value 1 at 1SPA. The aggregation at 1SPA computes $(1+0+1)/3 = 2/3$. Instead of one incinerator at 1SPA, the PHA end up with 2/3 at 1SPA and 1/3 at 9SPA. Having two incinerators lead to lower transportation cost, hence the value of z is bigger using PHA.

The next result is based on the feasible solution using GAMS. Let us consider parameters $f = 3$, $b = 500,000,000$, $h_s \in (50;80)$, $g_s \in (800;900)$ and $q_s \in (10,000;15,000)$. The PHA set values to 1 for 1SPA, 3SPA and 10SPA correctly. The rest of values are 0. This will lead to the same decision - to build incinerators 1SPA, 3SPA and 10SPA. Graphical results are shown in the figure 4.3.7 and the corresoponding values are in the table 4.3.4.

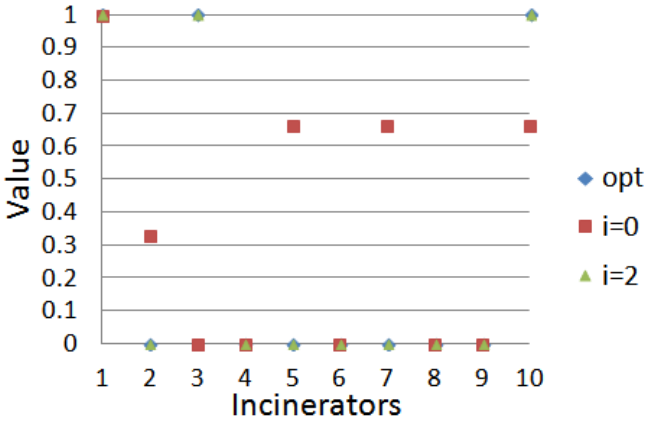

Figure 4.3.7: Comparison of the PHA result and the optimal result for the first-stage decision x_{is}.

	PHA		GAMS
is	i=0	i=2	opt
1SPA	1	1	1
2SPA	1/3	0	0
3SPA	0	1	1
4SPA	0	0	0
5SPA	2/3	0	0
6SPA	0	0	0
7SPA	2/3	0	0
8SPA	0	0	0
9SPA	0	0	0
10SPA	2/3	1	1

Table 4.3.4: Results after the 2nd iteration.

	z	r	i	solver
PHA	10,123,701,500	10^8	2	MINLP
GAMS	10,141,000,000			MIP

As you can see in the figure 4.3.7, PHA stopped after the second iteration, because the first-stage results for iterations $i = 1$ and $i = 2$ were the same. If we keep running the procedure a little bit longer, results will remain unchanged. Hence we can conclude, that the algorithm stabilizes after the second iteration in comparison to the Farmer's Problem 4.2, where we needed more iterations to obtain the convergence. In the Farmer's Problem, the first-stage decisions were real numbers. On the other hand the Main Problem has the binary first-stage decisions, which can take only 4 values $x_i \in \{0, 1/3, 2/3, 1\}$ after the aggregation.

It could be optimistic to expect an optimal result for the two-stage mixed integer stochastic program such as (4.4). Rather than optimal solution we end up with feasible solution. In my case, only settings for the figure 4.3.10 led to the optimal result. All the other models gave the feasible solution. The feasible solution can be considered as close to the optimal one, ie. suboptimal from the point of view of experts consulted.

The progressive hedging algorithm is not as fast as solvers in GAMS. But on the other hand, PHA is able to solve much bigger problems, that cannot be computed in GAMS. For instance, if the original model (4.4) would be SMINLP, then its computational complexity would lead to the decomposition and use of PHA, where subproblems would be smaller in size, but still be SMINLP. PHA can be programmed to run the computation in parallel to speed up the computation time. The other benefits of using the PHA lies in many modifications. I provided the modification for TS SMIP. For example, it is possible to compute lower bounds, change the penalty parameter r during the computation, etc. Roger J-B Wets and many other professors and scientists scientists focused their research on different aspects of PHA.

Graphical results are enclosed on the following pages on which the flow of the waste is captured. Pie charts represent the percentage amount of waste transported to landfills, incinerators and the penalty for not transported waste. Parameter settings are described in captions below each figure. The first-stage decisions x_i, values of the objective function z and the penalty restricted to the 1st scenario p_{ic,s_1} are written in tables below the corresponding figure. Changes in the parameter g_{s_1} with h_{s_1} fixed lead to the decision to build more than one incinerator. As we can see in the sequence of figures 4.3.9-4.3.13, 3 incinerators can cover the waste collection for the whole Czech Republic.

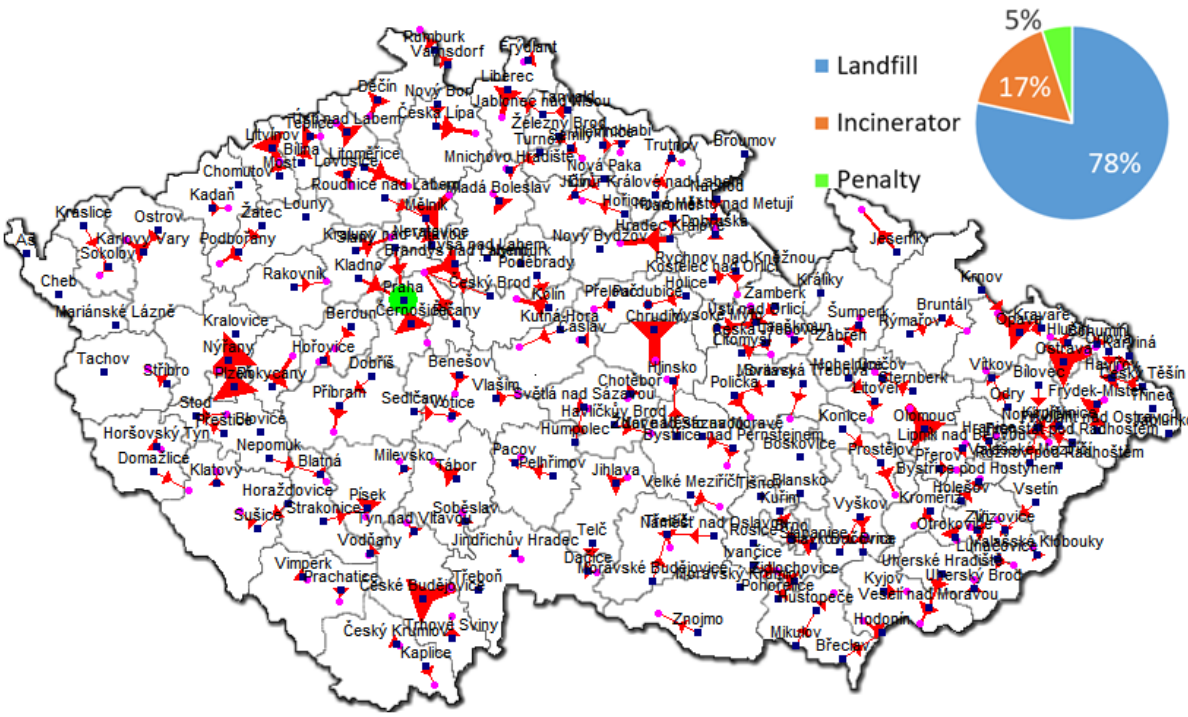

Figure 4.3.8: $f = 10$, $b = 500,000,000$, $h_{s_1} \in (50; 80)$, $g_{s_1} \in (200; 250)$, $q_{s_1} \in (40; 50)$.

x_i: 6SPA z: 194,890,500 Kč	p_{ic,s_1}: Rožnov pod Radhoštěm, Třeboň, Blansko, Boskovice, Moravský Krumlov, Rosice, Tišnov, Aš, Cheb, Mariánské Lázně, Broumov, Nový Bydžov, Tanvald, Jablunkov, Třinec, Králíky, Blovice, Kralovice, Tachov Lysá nad Labem, Nymburk, Louny.

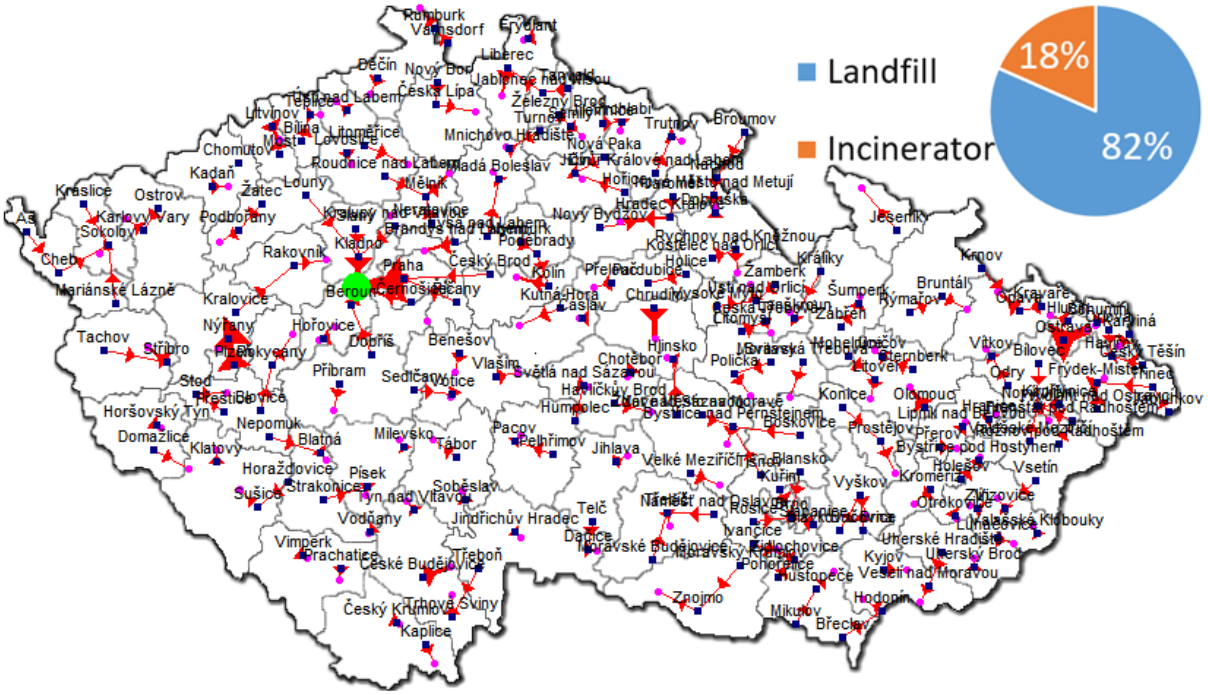

Figure 4.3.9: $f = 3$, $b = 500,000,000$, $h_{s_1} \in (50; 80)$, $g_{s_1} \in (100; 150)$, $q_{s_1} \in (10,000; 15,000)$.

x_i: 1SPA z: 455,423,700 Kč

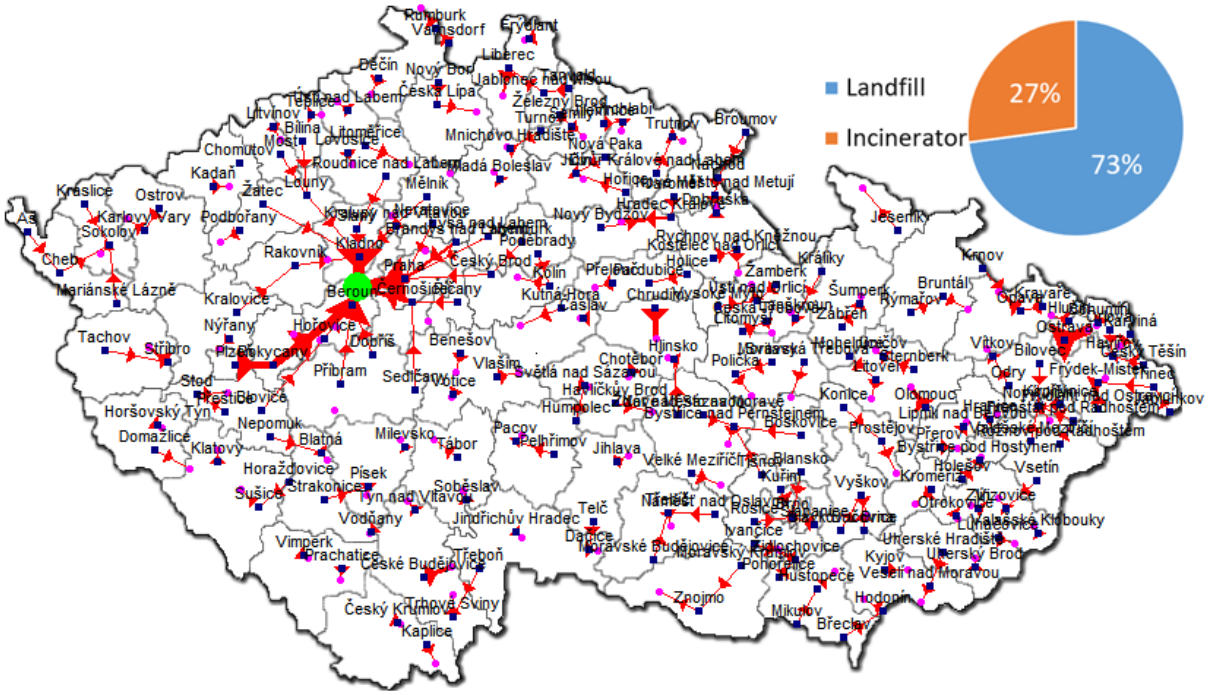

Figure 4.3.10: $f = 3$, $b = 500,000,000$, $h_{s_1} \in (50; 80)$, $g_{s_1} \in (200; 250)$, $q_{s_1} \in (10,000; 15,000)$.

x_i:	1SPA
z:	840,992,400 Kč

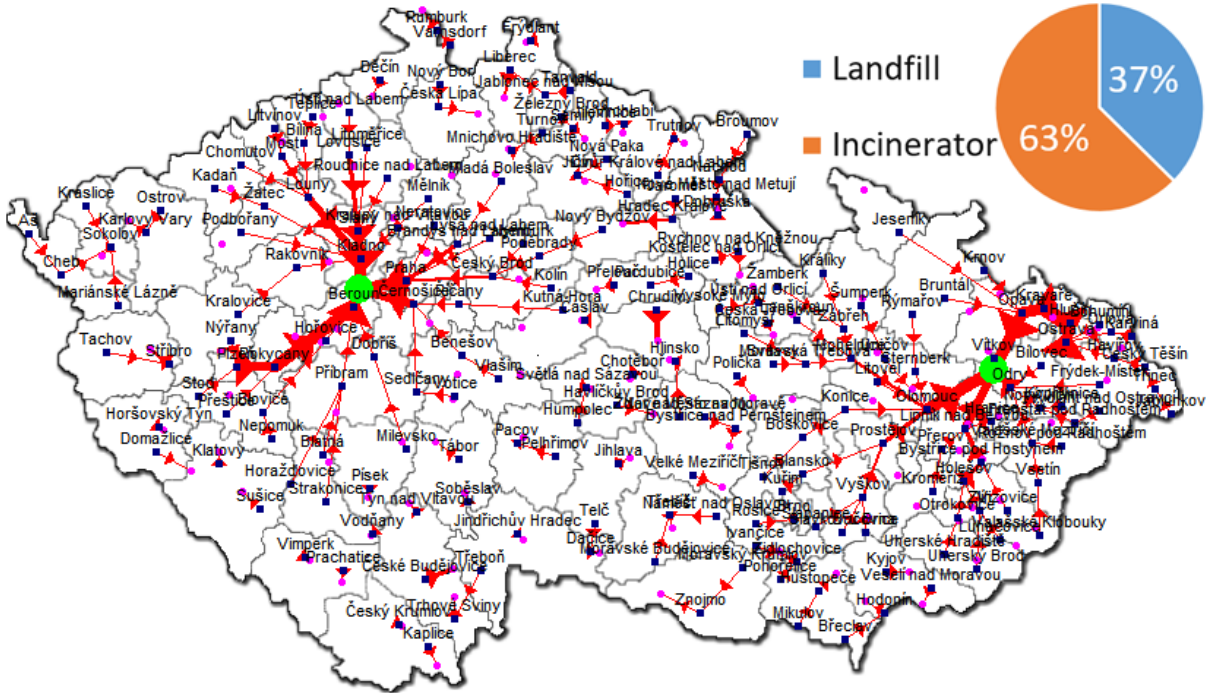

Figure 4.3.11: $f = 3$, $b = 500,000,000$, $h_{s_1} \in (50; 80)$, $g_{s_1} \in (300; 350)$, $q_{s_1} \in (10,000; 15,000)$.

x_i:	1SPA, 10SPA
z:	1,462,232,000 Kč

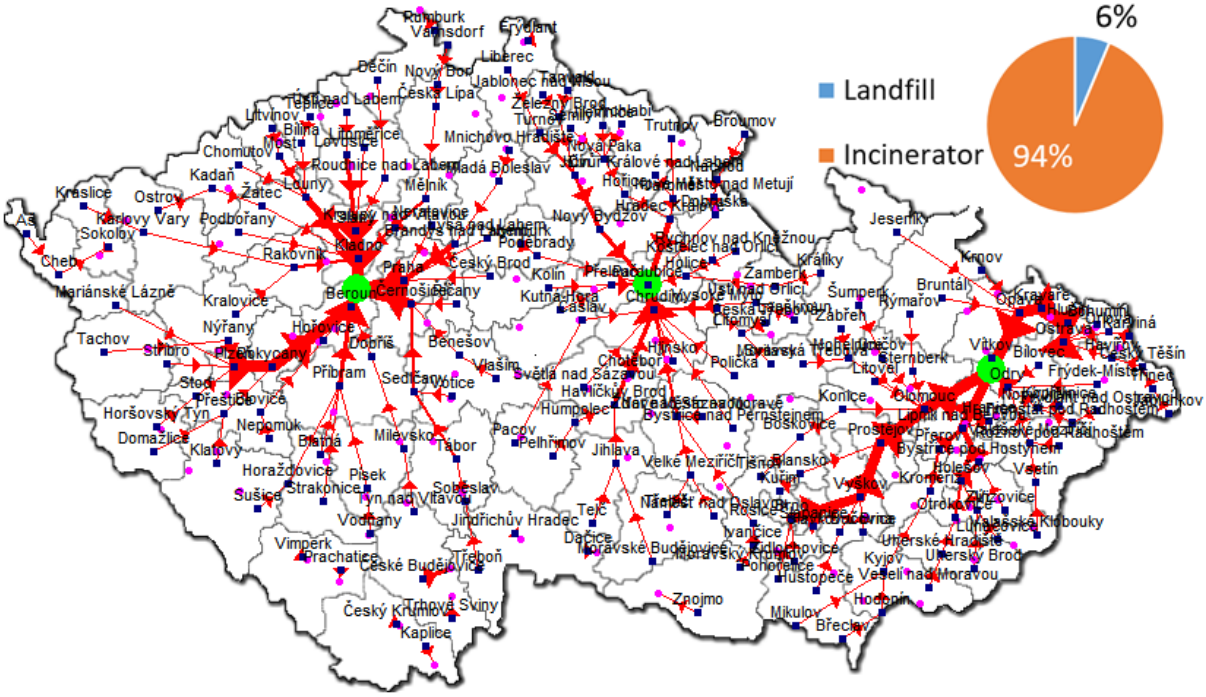

Figure 4.3.12: $f = 3$, $b = 500,000,000$, $h_{s_1} \in (50; 80)$, $g_{s_1} \in (400; 500)$, $q_{s_1} \in (10,000; 15,000)$.

x_i:	1SPA, 3SPA, 10SPA
z:	3,121,437,000 Kč

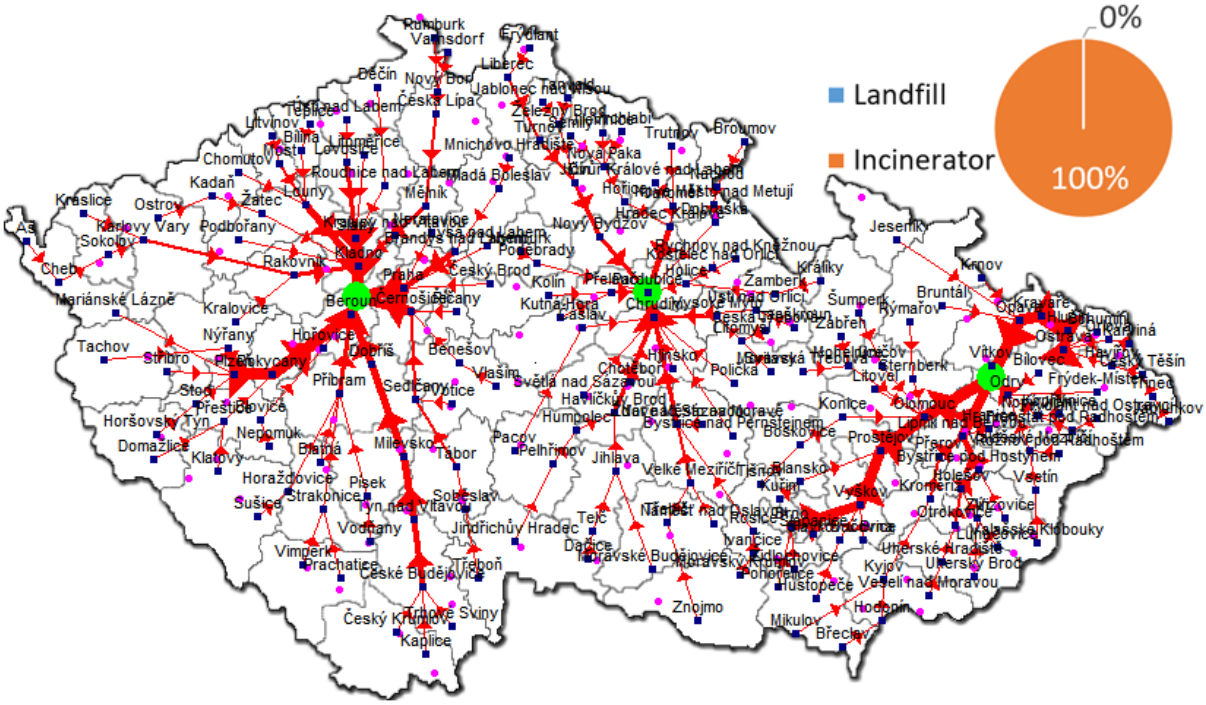

Figure 4.3.13: $f = 3$, $b = 500,000,000$, $h_{s_1} \in (50; 80)$, $g_{s_1} \in (800; 900)$, $q_{s_1} \in (10,000; 15,000)$.

x_i:	1SPA, 3SPA, 10SPA
z:	10,141,000,000 Kč

5. Conclusion

In the master's thesis I focused on the stochastic optimization and the implementation of the progressive hedging algorithm to the two-stage problems, including the mixed integer problem. The theory was described in the first three chapters, starting with Graph Theory 1, where I introduced the concept of network flow at the end.

The Optimization 2 started with a comparison between the deterministic and stochastic problems. The main emphasis was put on the stochastic programming involving the wait-and-see approach, here and-now-approach and the problems with more stages.

The Progressive Hedging Algorithm 3 was decribed as an alternative tool for solving the stochastic problems. The beginning of the chapter was devoted to the principle of scenario aggregation and afterwards the general form of the progressive hedging algorithm was introduced and modified to two-stage problems.

The principal part of thesis, Real-Data Applications 4, consists of 3 models, in which two of them use the real-world data of the collection of the municipal waste within the Czech Republic. Data were provided by the Institute of Process Engineering. In the Large Scale Problem 4.1, I used the probability weight between two variables and simulations to determine the behavior of the model. The Farmer's problem 4.2 served me for the implementation of the progressive hedging algorithm which was further used in the Main Problem 4.3. I built the two-stage mixed integer stochastic model and tested it on the small network. Then the real-world data were used and the model was computed by solvers in GAMS. Finally the modification of the progressive hedging algorithm for the mixed integer problem was provided and the computation was made using C++ programming language. Graphical results of network flows were modeled in AIMMS.

There are many further modifications of the progressive hedging algorithm. The algorithm used in thesis was in the serial form. If we want to speed up the computation, the progressive hedging algorithm can be implemented parallelly. Also the penalization parameter may not be fixed and may vary during the computation.

The code for the Main Problem is enclosed in appendices. All the other models described throughout the text are included on the CD. I hope this thesis will be useful for anyone, who is interested in the stochastic optimization, mainly in the progressive hedging algorithm.

Bibliography

[1] CHRISTOFIDES, Nicos: *Graph theory.* An algorithmic approach. London: Academic Press, 1975. Computer Science and Applied Mathematics. ISBN 0121743500.

[2] FOLTÝNEK, Tomáš and Jana DANNHOFEROVÁ: *Teorie grafů.* Brno: Mendel University in Brno, 2011. ISBN 9788073755003.

[3] GROSS, Jonathan L., Jay YELLEN and Ping ZHANG (eds.): *Handbook of graph theory.* 2nd ed. Boca Raton, Fla.: CRC Press, 2014. Discrete mathematics and its applications. ISBN 9781439880180.

[4] MÁLEK, M.: *Selected optimization models in logistics.* Bachelor's thesis. Brno: University of technology, Faculty of mechanical engineering, 2015. 71 p.

[5] RARDIN, R. L.: *Optimization in Operations Research.* Prentice Hall, 1998. ISBN 9780023984150.

[6] KALL, Peter. and Stein W. WALLACE: *Stochastic programming.* New York: Wiley, 1994. ISBN 0471951587.

[7] POPELA, P.: *Stochastic Programming*, Notes for Lectures, University of Malta, 2004.

[8] BIRGE, John R. and Francois. LOUVEAUX: *Introduction to stochastic programming.* New York: Springer, 1997. ISBN 0387982175.

[9] ROCKEFELLAR, R. T. Rockafellar and R. J. B. Wets: *Scenarios and policy aggregation in optimization under uncertainty.* In Mathematics of Operation Research, volume 16, pages 119-147. INFORMS, Linthicium, Maryland, 1991.

[10] AYDIN, Nezir, Alper MURAT and Boris S. MORDUKHOVICH: *Sampling-Based Progressive Hedging Algorithms in Two-Stage Stochastic Programming* [online]. 1-4 [cit. 2017-05-22]. Available at:`http://www.optimization-online.org/DB_FILE/2015/12/5232.pdf`

[11] KLIMEŠ, L.: *Stochastic programming algorithms.* Master's thesis. Brno: University of technology, Faculty of mechanical engineering, 2010.

[12] ZÉPHYR, L., P. LANG and B. F. LAMOND: *Adaptive monitoring of the progressive hedging penalty parameter in the context of reservoir systems management.* Québec: Faculté des sciences de l'administration, Université Laval, 2012. ISBN 9782895243755.

[13] CARPENTIER, P.L., M. GENDREAU and F. BASTIN: *Optimal scenario set partitioning for multistage stochastic programming with the progressive hedging algorithm* [online]. Québec: Faculté des sciences de l'administration, Université Laval, 2013. [cit. 2017-05-22]. Available at: `https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2013-55.pdf`

[14] POPELA, P.: *An object-oriented approach to multistage stoachastic programming models and algorithms*. Doctoral thesis. Faculty of mathematics and physics, Charles University, 1998.

[15] DANTZIG, G. B. and M. N. THAPA: *Linear programming.* New York: Springer, 2003. ISBN 0387986138.

[16] GADE, Dinakar, Gabriel HACKEBEIL, Sarah M. RYAN, Jean-Paul WATSON, Roger J.-B. WETS a David L. WOODRUFF: *Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs.* Mathematical Programming. 2016, 157(1), 47-67. DOI: 10.1007/s10107-016-1000-z. ISSN 0025-5610. Available at: `http://link.springer.com/10.1007/s10107-016-1000-z`

List of Used Symbols

Common shortcuts used in optimization

EEV	Expected objective function value for optimal solution of EV, 22
EO	Expected Objective Approach, 23
EV	Expected Value Approach, 22
EVPI	Expected Value of Perfect Information, 23
HN	Here-and-now Approach, 21
IS	Individual Scenario Approach, 22
LP	Linear Programming, 19
MIP	Mixed-Integer Programming, 20
MP	Mathematical Programming, 19
MS	Multi-stage Problem, 25
NLP	Non-Linear Programming, 20
PHA	Progressive Hedging Algorithm, 27
PMP	Parametric Mathematical Programming, 19
SP	Stochastic Programming, 20
TS	Two-stage Problem, 24
VSS	Value of Stochastic Solution, 23
WS	Wait-and-see Approach, 21

Graph Theory

$A = [a_{i,j}]$	adjacency matrix, 16
$A(G)$	set of arcs of the graph G, 15
$A_G^-(n)$	set of all arcs for which the final node is n, 15
$A_G^+(n)$	set of all arcs for which the initial node is n, 15
$a_{i,j} \in A, A'$	arc from the node i to the node j, set of arcs, subset of the set of arcs 14
$e_i \in E$	edge i, set of edges, 14
G, G'	graph, subgraph, 14
$M = [m_{i,j}]$	incidence matrix, 17
$n_i \in N, N'$	node i, set of nodes, subset of the set of nodes, 14
$N(G)$	set of nodes of the graph G, 15
$\mathbf{R}^+ = (0, +\infty)$	positive real numbers, 15
s, t	source, sink, 15
$v_i \in V$	vertice i, set of vertices, 14
$w(a)$	weight, 15
$x(a)$	flow, 15

Optimization and Progressive Hedging Algorithm

$\lozenge$	relation between the right hand side and the left hand side, $\leq, =, \geq$, 19
$\|\cdot\|$	Euclidean norm, 30
$\mathcal{A}_t$	partitioning, 28
A_i	scenario bundles, 28
$\mathbf{a}$	constant parameter, 19
$\mathbf{A}, \mathbf{b}$	coefficients of constraints, 19
a.s.	almost surely, 24
$\mathbf{c}^T$	coefficients of the objective function, 19

$C(\cdot)$	feasible set, 19
$\mathcal{C}$	admissible set, 29
$\mathcal{D}$	cartesian product of sets $\mathbf{R}^{n-l}$ and $\mathcal{Z}^l$, 20
$E[\cdot]$	expected value, 22
ϵ	termination parameter, 31
$\mathcal{E}$	space of all mappings, 29
$f(\mathbf{x})$	objective function, 19
$\mathcal{F} \subset \Omega$	family of events, 20
$\mathbf{g}(\mathbf{x})$	constraint functions, 20
$\mathcal{I}$	set of indices of variables, 19
$\mathcal{J}$	set of indices of constraints, 19
J	transforamtion, 29
$\mathbf{l}, \mathbf{u}$	lower bounds, upper bounds, 19
max	maximization, 41
min	minimization, 19
$\mathcal{N}$	implementable set, 29
$\mathbf{N}$	space of natural numbers, 19
ω	realization of random vector, 20
Ω	set of all possible realizations, 20
$\mathcal{P}$	probability distribution, 20
p_s	probability, , 29
$\mathcal{P}_s$	scenario subproblems, 31
π^i	termination condition, 31
$\mathbf{q}(\omega), \mathbf{T}(\omega), \mathbf{h}(\omega),$	second stage data, , 24
$\mathbf{R}^n$	n dimensional space of real numbers, 19
r	penalty parameter, 31
$s \in S$	scenario, set of all scenarios, 27
s.t.	subject to, 19
$var[\cdot]$	variance, 22
$\mathbf{W}^i(s)$	price system, 31
$\mathbf{X}(s) = (\mathbf{x}_1(s), ..., \mathbf{x}_N(s))$	policy, 27
$\hat{\mathbf{X}}(s) = (\hat{\mathbf{x}}_1(s), ..., \hat{\mathbf{x}}_N(s))$	aggregated policy, 29
$\mathbf{X}^*$	optimal scenario policy, 30
$\mathbf{x}, \mathbf{x}_1$	first stage decision variable, 19, 25
$\mathbf{x}^*$	optimal solution, 19
$\boldsymbol{\xi}$	random vector, 20
$\mathbf{y}$	second stage decision variable, 24
z	value of the objective function, 21
$\mathcal{Z}^l$	l dimensional space of integer numbers, 20
ζ	random variable, 22

References to used symbols in models in the Chapter 4

Model 4.1	Large-scale Problem, 33
Model 4.2	Farmer's Problem, 37
Model 4.3	Main Problem, 39

A GAMS

General **A**lgebraic **M**odeling **S**ystem, or simply GAMS, is a software for the mathematical programming and the optimization. The advantege lies in the notation similar to the algebraic notation. This makes GAMS easy to learn. GAMS consist of the language compiler and integrated high-performance solvers. It is built for large scale models, that makes them easily maintained and quickly adaptable to new situations.

Basic key words are listed in the table following table.

key word	what defines
`Set`	the set
`Scalar`	the scalar
`Parameter`	the parameter
`Variable`	the variable
`Equations`	the mathematical model
`Model`	name of the model
`Solve`	which model will be compiled; includes the direction
`Display`	display desired parameters, variables,...
`File`	the name of a file for an output
`Put`	starts writing into the output file

Table A.1: Basic key words.

In the thesis, GAMS is used to input data from MS Excel and to solve the MP. GAMS is independent of model and data. This means, we can include data from many different kind of sources using the GDX (**G**AMS **D**ata e**X**change) file format.

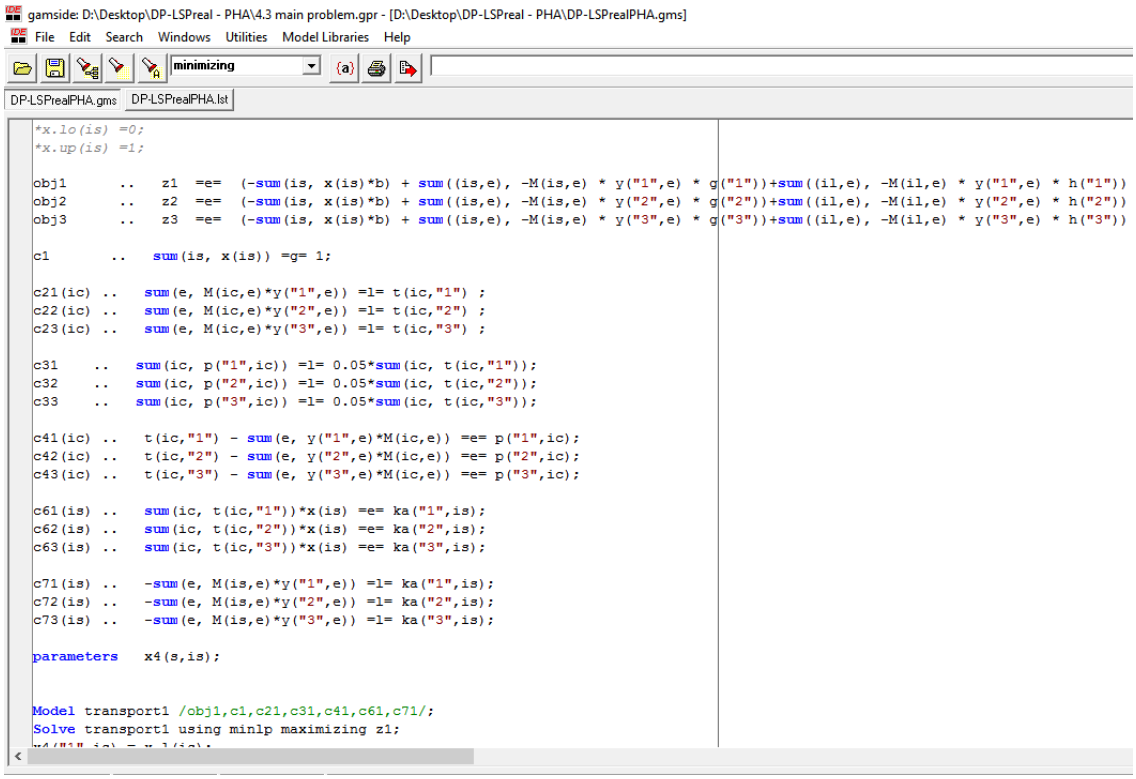

Figure A.1: Sample from the GAMS environment.

The sample of the GAMS environment is shown in the previous figure. Furthermore, I provide the part of the code in GAMS of PHA modification of the Main Problem. More information about GAMS can be found on the official website.

```
$TITLE Master's thesis - Main Problem - real data - PHA modification
$OFFLISTING
$EOLCOM //

sets i     index of nodes,
     k     index of coordinates,
     e     index of edges,
     s     index of scenarios,
     ic(i) index of cities,
     is(i) index of incinerators,
     il(i) index of landfills;

scalars f freight in coins per case using lorry /3/,
        b cost to build the incinerator /500000000/,
        r penalty parameter;

Parameters n(i,k)   coordinates of nodes,
           M(i,e)   incidence matrix,
           c(e)     tansportation cost,
           g(s)     zisk spalovna,
           q(s)     penale za neodvezeni,
           t(ic,s)  quantity of trash in cities,
           d(e)     distance by edges,
           Pr(s)    probability,
           h(s)     zisk skladka,
           xhat(is) aggregated x,
           wx(s,is) price system;

$onecho > tasks.txt
dset=i   rng=n,t!b2:b331                rdim=1
dset=k   rng=n,t!c1:d1                          cdim=1
dset=s   rng=n,t!h1:j1                          cdim=1
dset=ic  rng=n,t!b2:b207                rdim=1
dset=il  rng=n,t!b208:b321              rdim=1
dset=is  rng=n,t!b322:b331              rdim=1
par=n    rng=n,t!b1:d331                rdim=1   cdim=1
par=t    rng=n,t!g1:j331                rdim=1   cdim=1
par=d    rng=M,d!b336:cap337                     cdim=1
dset=e   rng=M,d!b1:cap1                         cdim=1
par=M    rng=M,d!a1:cap331              rdim=1   cdim=1
$offecho

$call   GDXXRW Import.xlsx trace=3 @tasks.txt
```

```
$GDXIN  Import.gdx
$LOAD   i k s e ic il is
$LOAD   n t d M
$GDXIN

h(s) = uniformint(50,80);
g(s) = uniformint(800,900);
q(s) = uniformint(10000,15000);
c(e) = d(e)*f;
Pr(s) = 1/card(s);
M(i,e) = -M(i,e);

$include InicPenalTermin.txt

Variables
       y(s,e)   amount of transported units
       z1       total profit - value of the objective function  s1
       z2       total profit - value of the objective function  s2
       z3       total profit - value of the objective function  s3
       p(s,ic)  number of penalty units
       x(is)    incinerators built
       ka(s,is) capacity of incinerator;

Positive Variables y, p, ka;
Binary variables x;

Equations
       obj1        objective function s1,
       obj2        objective function s2,
       obj3        objective function s3,
       c1          at least built 1 incinerator,
       c21(ic)     upper bound for y s1,
       c22(ic)     upper bound for y s2,
       c23(ic)     upper bound for y s3,
       c31         remove at least 95% of total waste s1,
       c32         remove at least 95% of total waste s3,
       c33         remove at least 95% of total waste s2,
       c41(i)      count penalty s1,
       c42(i)      count penalty s2,
       c43(i)      count penalty s3,
       c61(is)     capacity of incinerator s1,
       c62(is)     capacity of incinerator s2,
       c63(is)     capacity of incinerator s3,
       c71(is)     upper bound for incinerators,
       c72(is)     upper bound for incinerators,
       c73(is)     upper bound for incinerators;
```

```
obj1      ..   z1  =e=  (-sum(is, x(is)*b) + sum((is,e), -M(is,e) * y("1",e) *
        g("1"))+sum((il,e), -M(il,e) * y("1",e) * h("1")) - sum(e, c(e) *
y("1",e)) - sum(ic, p("1",ic) * q("1")) - sum(is, x(is) * wx("1",is))
- 0.5 * r * sum(is, (x(is) - xhat(is)) * (x(is) - xhat(is))))/100000 ;

obj2      ..   z2  =e=  (-sum(is, x(is)*b) + sum((is,e), -M(is,e) * y("2",e) *
g("2"))+sum((il,e), -M(il,e) * y("2",e) * h("2")) - sum(e, c(e) *
y("2",e)) - sum(ic, p("2",ic) * q("2")) - sum(is, x(is) * wx("2",is))
- 0.5 * r * sum(is, (x(is) - xhat(is)) * (x(is) - xhat(is))))/100000 ;

obj3      ..   z3  =e=  (-sum(is, x(is)*b) + sum((is,e), -M(is,e) * y("3",e) *
g("3"))+sum((il,e), -M(il,e) * y("3",e) * h("3")) - sum(e, c(e) *
y("3",e)) - sum(ic, p("3",ic) * q("3")) - sum(is, x(is) * wx("3",is))
- 0.5 * r * sum(is, (x(is) - xhat(is)) * (x(is) - xhat(is))))/100000 ;

c1        ..   sum(is, x(is)) =g= 1;

c21(ic) ..   sum(e, M(ic,e)*y("1",e)) =l= t(ic,"1") ;
c22(ic) ..   sum(e, M(ic,e)*y("2",e)) =l= t(ic,"2") ;
c23(ic) ..   sum(e, M(ic,e)*y("3",e)) =l= t(ic,"3") ;

c31     ..   sum(ic, p("1",ic)) =l= 0.05*sum(ic, t(ic,"1"));
c32     ..   sum(ic, p("2",ic)) =l= 0.05*sum(ic, t(ic,"2"));
c33     ..   sum(ic, p("3",ic)) =l= 0.05*sum(ic, t(ic,"3"));

c41(ic) ..   t(ic,"1") - sum(e, y("1",e)*M(ic,e)) =e= p("1",ic);
c42(ic) ..   t(ic,"2") - sum(e, y("2",e)*M(ic,e)) =e= p("2",ic);
c43(ic) ..   t(ic,"3") - sum(e, y("3",e)*M(ic,e)) =e= p("3",ic);

c61(is) ..   sum(ic, t(ic,"1"))*x(is) =e= ka("1",is);
c62(is) ..   sum(ic, t(ic,"2"))*x(is) =e= ka("2",is);
c63(is) ..   sum(ic, t(ic,"3"))*x(is) =e= ka("3",is);

c71(is) ..   -sum(e, M(is,e)*y("1",e)) =l= ka("1",is);
c72(is) ..   -sum(e, M(is,e)*y("2",e)) =l= ka("2",is);
c73(is) ..   -sum(e, M(is,e)*y("3",e)) =l= ka("3",is);

parameters   x4(s,is);


Model transport1 /obj1,c1,c21,c31,c41,c61,c71/;
Solve transport1 using minlp maximizing z1;
x4("1",is) = x.l(is);

Model transport2 /obj2,c1,c22,c32,c42,c62,c72/;
Solve transport2 using minlp maximizing z2;
x4("2",is) = x.l(is);
```

```
Model transport3 /obj3,c1,c23,c33,c43,c63,c73/;
Solve transport3 using minlp maximizing z3;
x4("3",is) = x.l(is);

file results_x1 /IS_x1.txt/;
put results_x1;
loop(s,
  loop(is,
    put x4(s,is):15:8;
  );
  put /;
);

file results_x2 /IS_x2.txt/;
put results_x2;

loop(e,
    put y.l("1",e):20:8;
    put y.l("2",e):20:8;
    put y.l("3",e):20:8;
    put /;
);
loop(ic,
    put p.l("1",ic):20:8;
    put p.l("2",ic):20:8;
    put p.l("3",ic):20:8;
    put /;
);
loop(is,
    put ka.l("1",is):20:8;
    put ka.l("2",is):20:8;
    put ka.l("3",is):20:8;
    put /;
);

file probab /probability.txt/;
put probab
loop(s,
  put Pr(s):15:8;
);

scalars pom_w1,pom_w2,pom_w3;
pom_w1 = - sum(is, x4("1",is) * wx("1",is)) - 0.5 * r * sum(is,
(x4("1",is) - xhat(is)) * (x4("1",is) - xhat(is)));
pom_w2 = - sum(is, x4("2",is) * wx("2",is)) - 0.5 * r * sum(is,
(x4("2",is) - xhat(is)) * (x4("2",is) - xhat(is)));
```

```
pom_w3 = - sum(is, x4("3",is) * wx("3",is)) - 0.5 * r * sum(is,
(x4("3",is) - xhat(is)) * (x4("3",is) - xhat(is)));

display x4,y.l,p.l,ka.l, pom_w1,pom_w2,pom_w3,z1.l,z2.l,z3.l,r;
```

B C++

C++ is the object-oriented programming language. For my purpose I used C++ in Visual Studio as a main software for the implementation of PHA. The code includes two classes `INITIALIZATION` and `INDIVIDUAL_SCENARIO` with the following functions.

```
class INITIALIZATION
{
public:
void PrintForeWord();          print the header
void GetValues();              obtain values of r and ϵ from user
void Set(double, double);      initialize price system, aggregated policy
void DataToTXT();              write data to .txt file as an input for GAMS
};

class INDIVIDUAL\_SCENARIO
{
public:
void OpenProgram(string);      compile GAMS
void Read_x1(string);          load the first stage decision x back to C++
void Read_x2(string);          load the second stage decision y back to C++
void Read_P(string);           load scenario probability back to C++
void Set_xhat();               compute aggregated policy x̂
void Set_yhat();               compute aggregated policy ŷ
void Update_w();               update price system
void TerminParam();            compute termination parameter π^i
void Update_tmp();             update temporary parameters
};
```

The C++ scheme of TS PHA is shown in the figure below. Afterwards, the code for the Main Problem is added.

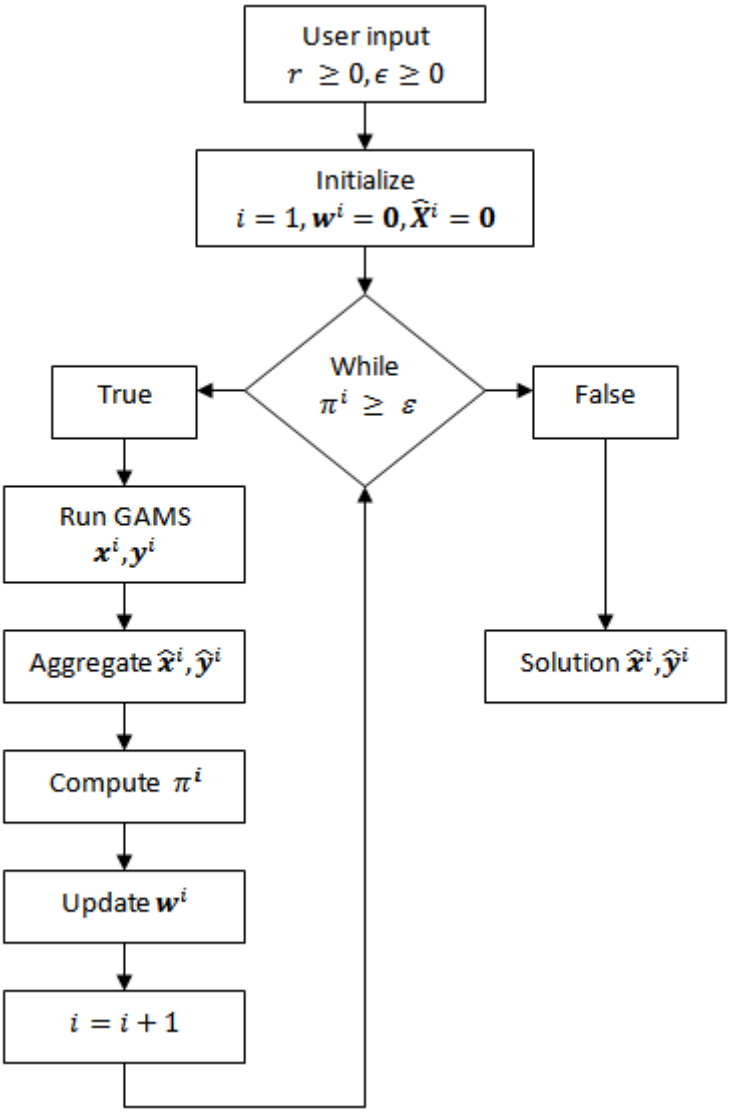

Figure B.1: Algorithm diagram.

```
#include <iostream>
#include <string>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <cmath>

using namespace std;

//GLOBAL VARIABLES
const int i = 10;    //no. of indexes of 1st stage decisions x(i)
const int j = 2285;    //no. of indexes of 2nd stage decisions y(j)
const int s = 3;    //no. of scenarios s
double wx[s][i];      //price system
double wy[s][j];       //price system
double yhat[s][j];  //weighted average for 2nd stage decisions
```

```
double xhat[i];     //weighted average for 1st stage decisions
double r = 0;           //penalty term
double e = 0;           //termination condition
double P[s];        //Probability of scenarios
double tmp_xhat[i]; //temporary parameters
double tmp_yhat[s][j];
double tmp_wx[s][i];
double tmp_wy[s][j];
double delta;       //termination parameter

class INITIALIZATION
{
double x=0, y;
public:
void PrintForeWord();
void GetValues();
void Set(double, double);
void DataToTXT();
};

void INITIALIZATION::PrintForeWord()
{
cout << "+--------------------------------------+" << endl;
cout << " Use PHA to solve TS SMINLP ." << endl;
cout << "+--------------------------------------+" << endl;
}

void INITIALIZATION::Set(double x, double y) //set initial parameters
{
r = x;
e = y;
for (int n = 0; n < i; n++)
{
for (int m = 0; m < s; m++)
{
wx[m][n] = 0;
}
}
for (int n = 0; n < j; n++)
{
for (int m = 0; m < s; m++)
{
wy[m][n] = 0;
}
}

for (int m = 0; m < i; m++)
```

```
{
xhat[m] = 0;
tmp_xhat[m] = xhat[m];
}

for (int n = 0; n < s; n++)
{
for (int m = 0; m < j; m++)
{
yhat[n][m] = 0;
tmp_yhat[n][m] = yhat[n][m];
}
}
DataToTXT();
}

void INITIALIZATION::DataToTXT()
{
ofstream myfile;
myfile.open("InicPenalTermin.txt");
myfile << "r = " << x << ";" << endl;
//myfile << "e = " << y << ";" << endl;
for (int m = 0; m<s; m++)
{
for (int n = 0; n < i; n++)
{
myfile<<"wx('"<< m+1<<"','"<<n+1<<"SPA')="<<wx[m][n]<<";"<<endl;
}
}

for (int m = 0; m < i; m++)
{
myfile << "xhat('"<< m + 1 << "SPA') = " << xhat[m] << ";" << endl;
}

myfile.close();

}

void INITIALIZATION::GetValues()
{
cout << "Set penalty parameter r: ";
cin >> x;
//cout << "Set termination parameter epsilon: ";
//cin >> y;
y = 0;
Set(x, y);
```

```
}

class INDIVIDUAL_SCENARIO
{
int row = 0, col = 0;
double x;
string line;
double x1[s][i] = { 0 };
double x2[s][j] = { 0 };
public:
void OpenProgram(string);
void Read_x1(string);
void Read_x2(string);
void Read_P(string);
void Set_xhat();
void Set_yhat();
void Update_w();
void TerminParam();
void Update_tmp();
};

void INDIVIDUAL_SCENARIO::OpenProgram(string filename)
{
system(filename.c_str());
}

void INDIVIDUAL_SCENARIO::Read_x1(string filename)
{
ifstream readfile;

readfile.open(filename); //read file
while (readfile.good())
{
while (getline(readfile, line))
{
istringstream stream(line);
col = 0;
while (stream >> x)
{
x1[row][col] = x;
col++;
}
row++;
}
}

cout << "First stage decisions: x1(s,i) = " << endl;
```

```
for (int i = 0; i<row; i++) //print file
{
for (int j = 0; j<col; j++)
{
cout << setw(6) << x1[i][j] << " ";
}
cout << endl;
}
cout << endl;
row = 0;
col = 0;
x = 0;
}

void INDIVIDUAL_SCENARIO::Read_x2(string filename)
{
ifstream readfile;
readfile.open(filename); //read file
while (readfile.good())
{
while (getline(readfile, line))
{
istringstream stream(line);
col = 0;
while (stream >> x)
{
x2[col][row] = x;
col++;
}
row++;
}
}

readfile.open(filename); //read file
while (readfile.good())
{
while (getline(readfile, line))
{
istringstream stream(line);
col = 0;
while (stream >> x)
{
x2[row][col] = x;
col++;
}
row++;
}
```

```
}


row = 0;
col = 0;
x = 0;
}

void INDIVIDUAL_SCENARIO::Read_P(string filename)
{
ifstream readfile;

readfile.open(filename); //read file
while (readfile.good())
{
while (getline(readfile, line))
{
istringstream stream(line);
col = 0;
while (stream >> x)
{
P[col] = x;
col++;
}
}
}

cout << "Probability of scenarios: P(s) = " << endl;

for (int j = 0; j<col; j++)
{
cout << setw(6) << P[j] << " ";
}
row = 0;
col = 0;
x = 0;
}

void INDIVIDUAL_SCENARIO::Set_xhat()
{
double sum[i] = { 0 };

for (int m = 0; m<i; m++) //initialization
{
xhat[m] = 0;
}
```

```
for (int m = 0; m<i; m++)  //set x_hat
{
for (int n = 0; n<s; n++)
{
sum[m] = P[n] * x1[n][m];
xhat[m] = xhat[m] + sum[m];
}
cout << endl;
}

cout << "x_hat(i) = " << endl;//print x_hat
for (int n = 0; n<i; n++)
{
cout << n+1 << "SPA    " << xhat[n] << endl;
}

}

void INDIVIDUAL_SCENARIO::Set_yhat()
{

for (int m = 0; m<s; m++)  //set y_hat
{
for (int n = 0; n<j; n++)
{
yhat[m][n] = x2[m][n];
}
cout << endl;
}

}

void INDIVIDUAL_SCENARIO::Update_w()
{

for (int m = 0; m<s; m++)
{
for (int n = 0; n<i; n++)
{
tmp_wx[m][n] = wx[m][n];   //set wx^j-1
}
}

for (int m = 0; m<s; m++)
{
for (int n = 0; n<j; n++)
{
```

```
tmp_wy[m][n] = wy[m][n];   //set wy^j-1
}
}

for (int m = 0; m<s; m++)
{
for (int n = 0; n<i; n++)
{
wx[m][n] = tmp_wx[m][n] + r*(x1[m][n] - xhat[n]);
}
}

cout << "Updated w: " << endl; //print
for (int m = 0; m<s; m++)
{
for (int n = 0; n<i; n++)
{
cout << setw(12) << wx[m][n];
}
cout << endl;
}
}

void INDIVIDUAL_SCENARIO::TerminParam()
{
double sum_1 = 0;
for (int n = 0; n < i; n++)
{
sum_1 = sum_1 + (tmp_xhat[n] - xhat[n])*(tmp_xhat[n] - xhat[n]);
}

double vect_1[s][j];
double sum_2[s] = { 0 };
double sum_3 = 0;
for (int n = 0; n<s; n++)
{
for (int m = 0; m<j; m++)
{
vect_1[n][m]=(tmp_yhat[n][m]-yhat[n][m])*(tmp_yhat[n][m]-yhat[n][m]);
sum_2[n] = sum_2[n] + vect_1[n][m];
}
sum_3 = sum_3 + sum_2[n];
}

double sum_4[s] = { 0 };
double sum_5 = { 0 };
for (int n = 0; n < s; n++)
```

```
{
for (int m = 0; m < i; m++)
{
sum_4[n] = sum_4[n] + (x1[n][m] - xhat[m])*(x1[n][m] - xhat[m]);
}
sum_5 = sum_5 + P[n] * sum_4[n];
}

delta = sum_1 + sum_3 + sum_5;
delta = sqrt(delta);

cout << "Delta = " << delta << endl;

}

void INDIVIDUAL_SCENARIO::Update_tmp()
{
for (int m = 0; m<i; m++)
{
tmp_xhat[m] = xhat[m];
}

for (int n = 0; n<s; n++)
{
for (int m = 0; m<j; m++)
{
tmp_yhat[n][m] = yhat[n][m];
}
}
}

int main()
{
double xhat_iter[i][250] = {0};
INITIALIZATION init;
INDIVIDUAL_SCENARIO IS;

init.PrintForeWord();

double dif = 0;
int iter = 0;
//init.GetValues();
init.Set(0,0);
IS.OpenProgram("gams DP-LSPrealPHA"); // initialization [1]
IS.Read_x1("IS_x1.txt");
IS.Read_x2("IS_x2.txt");
IS.Read_P("probability.txt");
```

```
IS.Set_xhat(); //aggregation [3]
for (int m = 0; m < i; m++)
{
dif = dif + abs(xhat_iter[m][iter] - xhat[m]);
xhat_iter[m][iter] = xhat[m];
}
cout << "dif: " << dif << endl;
IS.Set_yhat();
init.GetValues(); //set r, set e
IS.Update_w(); //Price update [4]
init.DataToTXT();
IS.Update_tmp();
iter += 1; //iteration update [2]

while(dif != 0) //Repeat until the results for x^i-1 and x^i are the same
{
IS.OpenProgram("gams DP-LSPrealPHA");
IS.Read_x1("IS_x1.txt");
IS.Read_x2("IS_x2.txt");
IS.Set_xhat();
IS.Set_yhat();
IS.Update_w();
init.DataToTXT();
IS.Update_tmp();

dif = 0;
for (int m = 0; m < i; m++)
{
dif = dif + abs(xhat_iter[m][iter-1] - xhat[m]);
xhat_iter[m][iter] = xhat[m];
}
cout << "dif: " << dif << endl;
iter++;
}

for (int m = 0; m < iter; m++)   //Print the results
{
cout << "iteration: " << m << "    " << endl;
cout << "+----------------+" << endl;
cout << setw(15) << "x_hat" << endl;
for (int n = 0; n < i; n++)
{
cout << n + 1 << "SPA" << setw(12) << xhat_iter[n][m];
cout << endl;
}
cout << endl;
}
```

```
ofstream myfile;    //Write results to the Results.txt file
myfile.open("Results.txt");

for (int m = 0; m < iter; m++)
{
myfile << m << "    ";
for (int n = 0; n < i; n++)
{
myfile << xhat_iter[n][m] << setw(12);
}
myfile << endl;
}

myfile.close();

cin.get();
cin.get();
return 0;
}
```

C AIMMS

The resulting file in GAMS is a .lst file with values. To be able to get a graphical result, we need a different software. For my purpose, the AIMMS (**A**dvanced **I**nteractive **M**ultidimensional **M**odeling **S**ystem) is suitable. AIMMS is designed for modeling and solving large-scale optimization problems. It is the combination of the integrated development evnironment, graphical user interface and numerical solvers. AIMMS covers the chain-supply management, logistics, automotive, chemical, energy and oil industry and it is used by large companies around the world such as Shell, Heineken, Lufthansa or PriceWaterhouseCoopers.

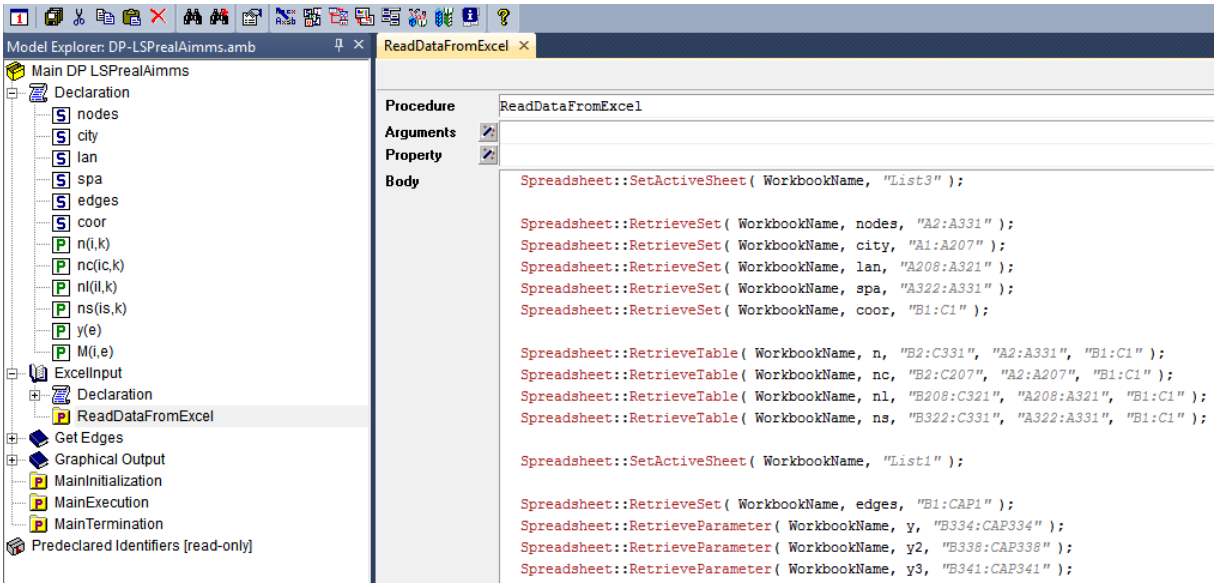

Figure C.1: AIMMS interface.

For better understanding I append the code of modeling the Main Problem. Further information about AIMMS can be found on the official website.

```
MAIN MODEL Main_DP_LSPrealAimms

  DECLARATION SECTION

    SET:
       identifier    :  nodes
       indices       :  i, j ;

    SET:
       identifier    :  city
       subset of     :  nodes
       index         :  ic ;

    SET:
       identifier    :  lan
       subset of     :  nodes
```

```
    index        :  il ;

  SET:
     identifier   :  spa
     subset of    :  nodes
     index        :  is ;

  SET:
     identifier   :  edges
     index        :  e ;

  SET:
     identifier   :  coor
     index        :  k ;

  PARAMETER:
     identifier   :  n
     index domain :  (i,k) ;

  PARAMETER:
     identifier   :  nc
     index domain :  (ic,k) ;

  PARAMETER:
     identifier   :  nl
     index domain :  (il,k) ;

  PARAMETER:
     identifier   :  ns
     index domain :  (is,k) ;

  PARAMETER:
     identifier   :  y
     index domain :  (e) ;

  PARAMETER:
     identifier   :  M
     index domain :  (i,e) ;

ENDSECTION  ;

SECTION ExcelInput


  DECLARATION SECTION

    STRING PARAMETER:
```

```
        identifier :  WorkbookName
        definition :  "proAimms.xlsx" ;

      PARAMETER:
        identifier :  ExcelStatus ;

      STRING PARAMETER:
        identifier :  ExcelErrorMessage ;

      ELEMENT PARAMETER:
        identifier :  err
        range      :  errh::PendingErrors ;

    ENDSECTION  ;

    PROCEDURE
      identifier :  ReadDataFromExcel
      body       :
        Spreadsheet::SetActiveSheet( WorkbookName, "List3" );

        Spreadsheet::RetrieveSet( WorkbookName, nodes, "A2:A331" );
        Spreadsheet::RetrieveSet( WorkbookName, city, "A1:A207" );
        Spreadsheet::RetrieveSet( WorkbookName, lan, "A208:A321" );
        Spreadsheet::RetrieveSet( WorkbookName, spa, "A322:A331" );
        Spreadsheet::RetrieveSet( WorkbookName, coor, "B1:C1" );

        Spreadsheet::RetrieveTable( WorkbookName, n, "B2:C331",
"A2:A331", "B1:C1" );
        Spreadsheet::RetrieveTable( WorkbookName, nc, "B2:C207",
"A2:A207", "B1:C1" );
        Spreadsheet::RetrieveTable( WorkbookName, nl, "B208:C321",
"A208:A321", "B1:C1" );
        Spreadsheet::RetrieveTable( WorkbookName, ns, "B322:C331",
"A322:A331", "B1:C1" );

        Spreadsheet::SetActiveSheet( WorkbookName, "List1" );

        Spreadsheet::RetrieveSet( WorkbookName, edges, "B1:CAP1" );
        Spreadsheet::RetrieveParameter( WorkbookName, y, "B334:CAP334" );
        Spreadsheet::RetrieveParameter( WorkbookName, y2, "B338:CAP338" );
        Spreadsheet::RetrieveParameter( WorkbookName, y3, "B341:CAP341" );
        Spreadsheet::RetrieveParameter( WorkbookName, y4, "B344:CAP344" );
        Spreadsheet::RetrieveParameter( WorkbookName, y5, "B347:CAP347" );
        Spreadsheet::RetrieveParameter( WorkbookName, y6, "B350:CAP350" );
        Spreadsheet::RetrieveParameter( WorkbookName, y7, "B353:CAP353" );
        Spreadsheet::RetrieveTable( WorkbookName, M, "B2:CAP331",
"A2:A331", "B1:CAP1" );
```

```
  ENDPROCEDURE  ;

ENDSECTION ExcelInput ;

SECTION Get_Edges


  DECLARATION SECTION

    PARAMETER:
       identifier   :  y_pom
       index domain :  (e,i,j) ;

    PARAMETER:
       identifier   :  x
       index domain :  (i,j) ;

    PARAMETER:
       identifier   :  x2
       index domain :  (i,j) ;

    PARAMETER:
       identifier   :  x3
       index domain :  (i,j) ;

    PARAMETER:
       identifier   :  x4
       index domain :  (i,j) ;

    PARAMETER:
       identifier   :  x5
       index domain :  (i,j) ;

    PARAMETER:
       identifier   :  x6
       index domain :  (i,j) ;

    PARAMETER:
       identifier   :  x7
       index domain :  (i,j) ;

  ENDSECTION  ;

  PROCEDURE
    identifier :  GetEdges
    body       :
```

```
y_pom(e,i,j):= 0;
for e in edges do
for i in nodes do
if M(i,e) = -1 then
for j in nodes do
if M(j,e) = 1 then
y_pom(e,i,j):= y(e); break; endif; endfor; break; endif;

endfor;

endfor;

x(i,j):=sum(e,y_pom(e,i,j));

y_pom(e,i,j):= 0;
for e in edges do
for i in nodes do
if M(i,e) = -1 then
for j in nodes do
if M(j,e) = 1 then
y_pom(e,i,j):= y2(e); break; endif; endfor; break; endif;

endfor;

endfor;

x2(i,j):=sum(e,y_pom(e,i,j));

y_pom(e,i,j):= 0;
for e in edges do
for i in nodes do
if M(i,e) = -1 then
for j in nodes do
if M(j,e) = 1 then
y_pom(e,i,j):= y3(e); break; endif; endfor; break; endif;

endfor;

endfor;

x3(i,j):=sum(e,y_pom(e,i,j));

y_pom(e,i,j):= 0;
for e in edges do
for i in nodes do
if M(i,e) = -1 then
for j in nodes do
```

```
if M(j,e) = 1 then
y_pom(e,i,j):= y4(e); break; endif; endfor; break; endif;

endfor;

endfor;

x4(i,j):=sum(e,y_pom(e,i,j));

y_pom(e,i,j):= 0;
for e in edges do
for i in nodes do
if M(i,e) = -1 then
for j in nodes do
if M(j,e) = 1 then
y_pom(e,i,j):= y5(e); break; endif; endfor; break; endif;

endfor;

endfor;

x5(i,j):=sum(e,y_pom(e,i,j));

y_pom(e,i,j):= 0;
for e in edges do
for i in nodes do
if M(i,e) = -1 then
for j in nodes do
if M(j,e) = 1 then
y_pom(e,i,j):= y6(e); break; endif; endfor; break; endif;

endfor;

endfor;

x6(i,j):=sum(e,y_pom(e,i,j));

y_pom(e,i,j):= 0;
for e in edges do
for i in nodes do
if M(i,e) = -1 then
for j in nodes do
if M(j,e) = 1 then
y_pom(e,i,j):= y7(e); break; endif; endfor; break; endif;

endfor;
```

```
      endfor;

      x7(i,j):=sum(e,y_pom(e,i,j));

  ENDPROCEDURE  ;

ENDSECTION Get_Edges ;

SECTION Graphical_Output


  DECLARATION SECTION

    SET:
       identifier   :  spav
       subset of    :  spa
       index        :  iss ;

    SET:
       identifier   :  spav2
       subset of    :  spa
       index        :  iss2 ;

    PARAMETER:
       identifier   :  y2
       index domain :  (e) ;

    PARAMETER:
       identifier   :  y3
       index domain :  (e) ;

    PARAMETER:
       identifier   :  y4
       index domain :  (e) ;

    PARAMETER:
       identifier   :  y5
       index domain :  (e) ;

    PARAMETER:
       identifier   :  y6
       index domain :  (e) ;

    PARAMETER:
       identifier   :  y7
       index domain :  (e) ;
```

```
      SET:
         identifier   :  spav3
         subset of    :  spa
         index        :  iss3 ;

      SET:
         identifier   :  spav4
         subset of    :  spa
         index        :  iss4 ;

      SET:
         identifier   :  spav5
         subset of    :  spa
         index        :  iss5 ;

      SET:
         identifier   :  spav6
         subset of    :  spa
         index        :  iss6 ;

      SET:
         identifier   :  spav7
         subset of    :  spa
         index        :  iss7 ;

    ENDSECTION  ;

  ENDSECTION Graphical_Output ;

  PROCEDURE
    identifier :  MainInitialization

  ENDPROCEDURE  ;

  PROCEDURE
    identifier :  MainExecution

  ENDPROCEDURE  ;

  PROCEDURE
    identifier :  MainTermination
    body       :
      return DataManagementExit();

  ENDPROCEDURE  ;

ENDMODEL Main_DP_LSPrealAimms ;
```